Practice exam: Databases and Data Management

Duration: 2 hours. Answer all questions. The questions follow the lecture order. The exam is designed for online answering, so no drawing is required and no relational algebra notation needs to be typed by the student.

Questions

Question 1 (Mark the correct choices, estimated time 6 minutes, difficulty: easy) Which statements correctly describe a database and a database management system?

A. A DBMS is software used to define, store, query, and manage databases.

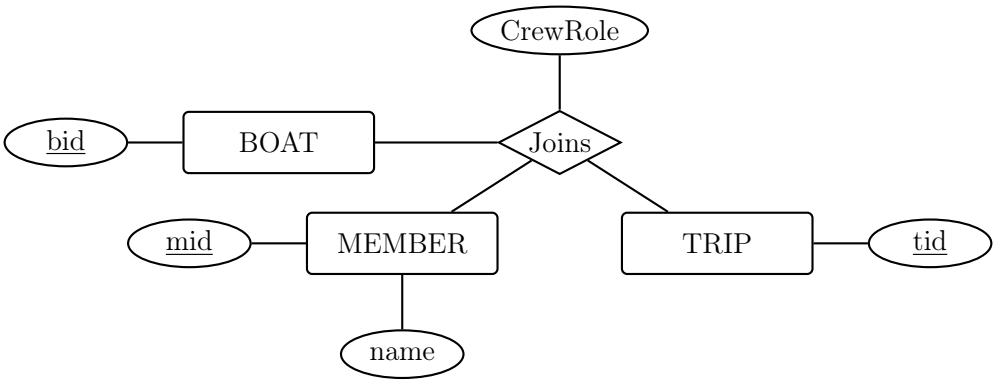
B. A database is an organized collection of related data.

C. A DBMS and a database are exactly the same thing.

D. A plain text file containing a shopping list is always a database in the DBMS sense.

Answer. Correct answers: A,B.
Explanation: A DBMS is software for creating and managing databases, while a database is the stored collection of data itself. They are related, but not identical.

Question 2 (Mark the correct choices, estimated time 8 minutes, difficulty: easy) The figure below is adapted from the ER chapter.



A. This ER design is wrong, because relationship sets cannot have attributes.

B. This ER design is wrong, because `CrewRole` should instead be copied as an attribute of both `MEMBER` and `TRIP`.

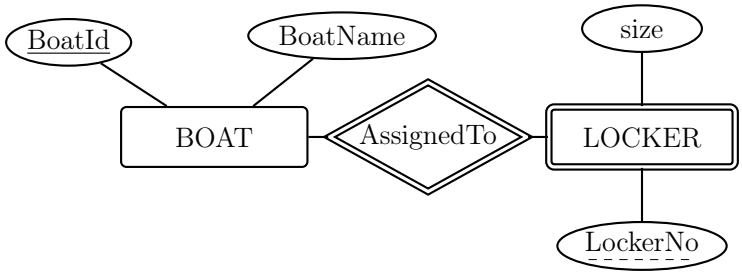
C. In a database based on this design, one tuple would represent one `mid, tid, bid` participation triple, so the same member joining the same trip on the same boat and the same role again would not be separately represented unless the design is extended.

D. Every trip has exactly one crew role.

E. The ternary relationship can always be replaced by the three binary relationships `MEMBER-TRIP`, `MEMBER-BOAT`, and `TRIP-BOAT` without loss of information.

Answer. Correct answer: C.
Explanation: Relationship sets can have attributes, so A is false. Assigning `CrewRole` to entity sets such as `MEMBER` and `TRIP` would distort the meaning, so B is false. C is correct because the relationship instance is based on the participating entity keys together, namely `mid, tid, bid`. D is false because no cardinality constraint says that every trip has exactly one crew role. E is false because a ternary relationship is not in general equivalent to three binary relationships: the ternary relationship stores the three-way association as one fact.

Question 3 (multiple choice, mark the correct choices (if any), estimated time 10 minutes, difficulty: medium) The figure below is adapted from the weak entity example in the notes.



Mark all correct statements.

A. `LOCKER` is a weak entity set and every locker is attached to exactly one boat.

B. `AssignedTo` is an identifying relationship.

C. `LockerNo` alone is sufficient to identify a locker globally in the database.

D. The key of the weak entity is understood together with the owner key.

E. If a boat is removed, the related weak-entity information may naturally disappear as well.

Answer. Correct answers: A, B, D, E.
Explanation: `LOCKER` is shown as a weak entity, `AssignedTo` is an identifying relationship, and the partial key `LockerNo` is interpreted together with the key of the owner entity. Weak entities depend on their owner semantically.

Question 4 (multiple-choice, mark all correct answers, estimated time 8 minutes, difficulty: easy) Consider the relational design

$$\texttt{BOAT}(\underline{\texttt{boatid}}, \texttt{bname}, \texttt{btype}, \texttt{homeport}),$$

$$\texttt{SKIPPER}(\underline{\texttt{sid}}, \texttt{sname}, \texttt{license}),$$

$$\texttt{CHARTERED}(\underline{\texttt{boatid}}, \texttt{sid}, \texttt{fee}, \texttt{cdate}).$$

Which of the following statements are correct? Mark all correct answers.

A. `CHARTERED` represents a relationship between `BOAT` and `SKIPPER`.

B. `fee` and `cdate` are attributes of the relationship recorded in `CHARTERED`.

C. `boatid` should be removed from `CHARTERED` because it already appears in `BOAT`.

D. This design can represent that one skipper chartered several boats.

E. This design can represent that the same boat was chartered in several different recorded transactions.

Answer. Correct answers: A, B, D.
Explanation: `CHARTERED` is the relationship relation, and `fee` and `cdate` describe that relationship. The foreign keys must remain in the relation. Because `boatid` is underlined in `CHARTERED`, the same boat can occur only once there.

Question 5 (multiple choice, several correct answers, estimated time 8 minutes, difficulty: medium) Mark all correct statements about the relational model and integrity constraints.

A. All tuples in a relation follow the same schema.

B. In the relational model, tuple order is part of the data semantics.

C. A key attribute may take the value `NULL` if all other key values are distinct.

D. A foreign key may refer to a key in another relation.

E. A foreign key may refer to a key in the same relation.

Answer. Correct answers: A, D, E.
Explanation: All tuples in one relation share the same attributes. Tuple order is not semantically important in the relational model. Keys cannot contain `NULL`. Foreign keys can refer to another relation or to the same relation.

Question 6 (SQL query writing, estimated time 11 minutes, difficulty: medium) Consider the SQLite schema

$$\texttt{FLIGHT}(\texttt{fid}, \texttt{fromcity}, \texttt{tocity}, \texttt{airplaneType}, \texttt{airline})$$

Write an SQLite query that returns all pairs of flight identifiers `fid` and the corresponding airlines for one-stop flight connections from `'Montevideo'` to `'Helsinki'`. A one-stop connection means that the first flight goes from `'Montevideo'` to some intermediate city, and the second flight goes from that same intermediate city to `'Helsinki'`.

Answer. One correct SQLite query is:

```
SELECT F1.fid, F1.airline, F2.fid, F2.airline
FROM FLIGHT AS F1, FLIGHT AS F2
WHERE F1.tocity = F2.fromcity
  AND F1.fromcity = 'Montevideo'
  AND F2.tocity = 'Helsinki';
```

Explanation: The query uses the `FLIGHT` relation twice. `F1` represents the first leg from Montevideo to an intermediate city, and `F2` represents the second leg from that same intermediate city to Helsinki. The `SELECT` clause returns both flight identifiers together with the airline of each leg.

Question 7 (SQL query writing, estimated time 11 minutes, difficulty: hard) Consider the SQLite schema

$$\texttt{MEMBER}(\texttt{mid}, \texttt{mname}, \texttt{city}, \texttt{mtype}, \texttt{region})$$

$$\texttt{BOOKING}(\texttt{bid}, \texttt{mid}, \texttt{byear}, \texttt{amount}, \texttt{status})$$

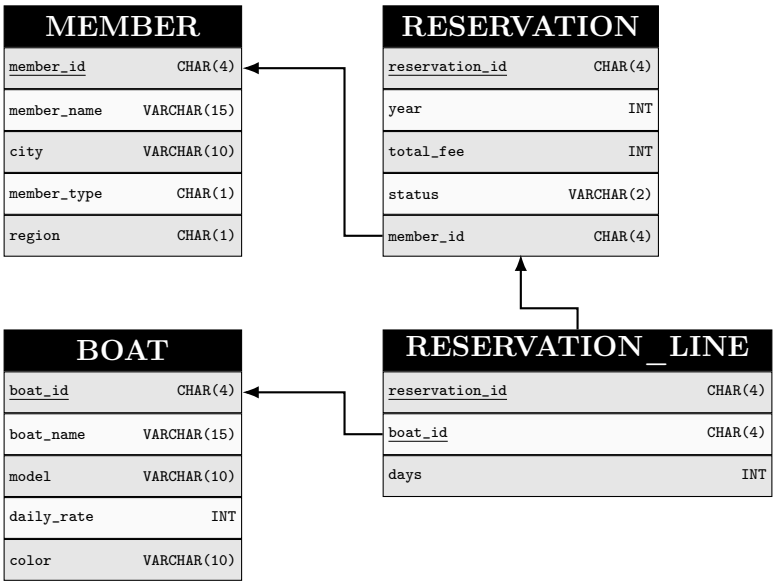
Write an SQLite query that returns the `mname` and `mid` of the member or members who have made the largest number of bookings in the database.

Answer. One correct SQLite query is:

```
SELECT MEMBER.mname, MEMBER.mid
FROM MEMBER, BOOKING
WHERE MEMBER.mid = BOOKING.mid
GROUP BY MEMBER.mid, MEMBER.mname
HAVING COUNT(*) = (
    SELECT MAX(cnt)
    FROM (
        SELECT COUNT(*) AS cnt
        FROM BOOKING
        GROUP BY mid
    )
);
```

Explanation: The outer query joins `MEMBER` and `BOOKING`, groups the rows by member, and counts how many bookings each member has. The subquery computes the booking count for each member and then finds the largest such count. The `HAVING` clause keeps exactly those member groups whose count matches that maximum. If several members are tied for the largest number of bookings, they are all returned.

Question 8 (single-choice multiple choice, estimated time 9 minutes, difficulty: medium) Use the schema figure below from the SQL chapter.



Which query correctly returns the distinct cities of members that have at least one reservation?

A. `SELECT city FROM MEMBER, RESERVATION;`

B. `SELECT DISTINCT m.city FROM MEMBER AS m JOIN RESERVATION AS r ON m.member_id = r.member_id;`

C. `SELECT DISTINCT city FROM RESERVATION;`

D. `SELECT city FROM MEMBER WHERE reservation_id IS NOT NULL;`

Answer. Correct answer: B.
Explanation: The join links each reservation to its member, and `DISTINCT` removes duplicate city names.

Question 9 (multiple-choice, mark all correct answers, estimated time 10 minutes, difficulty: medium) Consider the SQLite relation

$$\texttt{BOOKING}(\texttt{bid}, \texttt{mid}, \texttt{byear}, \texttt{amount}, \texttt{status})$$

Which of the following SQLite statements correctly return each `mid` together with the total sum of `amount`, but only for members whose total exceeds `1500`? Mark all correct answers.

A.
```
SELECT mid, SUM(amount) AS total_amount
FROM BOOKING
WHERE SUM(amount) > 1500
GROUP BY mid;
```

B.
```
SELECT mid, SUM(amount) AS total_amount
FROM BOOKING
GROUP BY mid
HAVING SUM(amount) > 1500;
```

C.
```
SELECT mid, SUM(amount) AS total_amount
FROM BOOKING
GROUP BY mid
HAVING total_amount > 1500;
```

D.
```
SELECT DISTINCT mid, amount AS total_amount
FROM BOOKING
WHERE amount > 1500;
```

Answer. Correct answers: B, C.
Explanation: The rows must first be grouped by `mid`, then filtered based on the aggregate total using `HAVING`. Option A incorrectly uses an aggregate in `WHERE`. Option D filters individual rows rather than grouped totals.

Question 10 (multiple-choice, mark all correct answers, estimated time 8 minutes, difficulty: easy) Consider the ACID properties of transactions. Which statements are correct? Mark all correct answers.

A. A = Atomicity: A transaction either happens fully or not at all.

B. C = Consistency: If a transaction starts in a consistent state and is correct, it should leave the database in a consistent state.

C. I = Isolation: Concurrent transactions should behave as if executed one after another in some serial order.

D. D = Durability: After commit, the effects are permanent even if a failure occurs soon after.

Answer. Correct answers: A, B, C, D.
Explanation: These are the standard meanings of the ACID properties.

Question 11 (matching, estimated time 12 minutes, difficulty: hard) Consider the following two relations.

`Person(un, city, age)`

un	**city**	**age**
Kim	Abo	31
Ina	Ulm	22
Ada	NYC	43
Rui	Sian	43

`Follows(un, un1)`

un	**un1**
Kim	Ada
Ada	Rui

Match each relational algebra expression on the left with the correct result on the right. You only need to write the matching pairs, for example `1-B, 2-D, ...`. Do not type any new relational algebra expression.

Left		**Right**	
1	$\sigma_{age=43}(\texttt{Person} \bowtie \texttt{Follows})$	A	$\{(\texttt{Ada}, \texttt{NYC}, 43, \texttt{Rui})\}$
2	$\pi_{un}(\sigma_{city='Ulm'}(\texttt{Person}))$	B	$\{(\texttt{Ina})\}$
3	$\sigma_{age=43}(\texttt{Person})$	C	$\{(\texttt{Ada}, \texttt{NYC}, 43), (\texttt{Rui}, \texttt{Sian}, 43)\}$
4	$\pi_{un}(\texttt{Follows})$	D	$\{(\texttt{Kim}), (\texttt{Ada})\}$
		E	$\{(\texttt{Kim}, \texttt{Ada}), (\texttt{Ada}, \texttt{Rui})\}$
		F	$\{(\texttt{Ada}), (\texttt{Rui})\}$
		G	$\{(\texttt{Kim}, \texttt{Abo}, 31), (\texttt{Ina}, \texttt{Ulm}, 22)\}$
		H	$\{(\texttt{Kim}, \texttt{Abo}, 31, \texttt{Ada}), (\texttt{Ada}, \texttt{NYC}, 43, \texttt{Rui})\}$

Answer. Correct answer: `1-A, 2-B, 3-C, 4-D`.
Explanation: Expression 1 first performs the natural join of `Person` and `Follows` on the common attribute `un`. This gives the tuples $(\texttt{Kim}, \texttt{Abo}, 31, \texttt{Ada})$ and $(\texttt{Ada}, \texttt{NYC}, 43, \texttt{Rui})$. Selecting `age=43` keeps only the second tuple, so 1 matches A. Expression 2 selects the person whose city is `Ulm`, namely Ina, and then projects the username, so it matches B. Expression 3 selects all tuples from `Person` with age 43, namely Ada and Rui, so it

matches C. Expression 4 projects the `un` column of `Follows`, so it returns Kim and Ada, which matches D.

The other options are distractors. E is the original `Follows` relation. F is what would be obtained by projecting `un1` from `Follows`. G is a different subset of `Person`. H is the full natural join result before the selection on age.

Question 12 (multiple-choice, mark all correct answers, estimated time 10 minutes, difficulty: hard) Let the relation schema be

$$T(A, B, C, D, E, F)$$

with functional dependencies

$$AB \to F, \qquad C \to D, \qquad C \to A, \qquad E \to ABCDEF.$$

Which of the following decompositions are lossless-join decompositions into BCNF? Mark all correct answers.

A. `EC`, `CD`, `ABEF`

B. `ABF`, `CD`, `ABCE`

C. `EC`, `CAD`, `BEF`

D. `ABF`, `CD`, `EF`

Answer. Correct answers: C.
Explanation:

- **A:** `ABEF` is not in BCNF because $AB \to F$ and AB is not a superkey of `ABEF`.
- **B:** `ABCE` is not in BCNF because $C \to A$ holds but C is not a superkey of `ABCE`.
- **C:** `EC` is in BCNF, `CAD` is in BCNF because $C \to AD$, and `BEF` is in BCNF because $E \to BF$. The decomposition is lossless-join.
- **D:** Even if each component relation is in BCNF, the decomposition is not lossless-join.