4-hours written exam in "Databases and Information Systems"—DIS, 2022

Datalogisk Institut, Københavns Universitet (DIKU)
Date: June 23, 2022

Preamble

Solution

Disclaimer: In the following, we present solution sketches for the various questions in the exam. These solution sketches are provided only as a reference, and may lack details that we would expect in a complete answer to the exam. Moreover, some of the questions may admit more than one correct solution, but even in such cases only one solution sketch is provided for brevity. Solution sketches are colored for visibility.

Note, in addition, that the evaluation of the exam takes into account our expectations regarding solutions, the actual formulations provided, the weights of various questions, but also and most importantly the overall evaluation of the exam assignment as a whole. This evaluation is performed by both internal and external examiners and grades are finally provided by discussion and consensus. As such, it is not advised to reason about final grades based on this document.

The exam will be evaluated on the 7-point grading scale with external grading, as announced in the course description.

- Your answers must be provided in English.
- Hand-ins for this exam must be individual, so cooperation with others in preparing a solution is strictly forbidden.
- You are allowed to use books, notes and other written material from the course. If you use any other sources, they must be cited appropriately.

Errors and Ambiguities

If you find any errors or ambiguities in the exam text, you should clearly state your assumptions in answering the corresponding questions. Some of the questions may not have a single correct answer, so recall that ambiguities could be intentional.

Expectations regarding question weights

Questions carry indicative weights. The weights will be used during evaluation to prioritize question answers towards grading; however, recall that the exam is still evaluated as a whole. In other words, we provide weights only as an indication so that you can prioritize your time during the exam, if you need to. You cannot assume that weights will be divided equally among subquestions.

The following table summarizes the questions in this exam and their weights.

	Question	Weight
Q1	Entity–Relationship model	15%
Q2	Functional dependencies, keys, BCNF	10%
Q3	Relational calculus	25%
Q4	Relational algebra	25%
Q5	SQL	25%

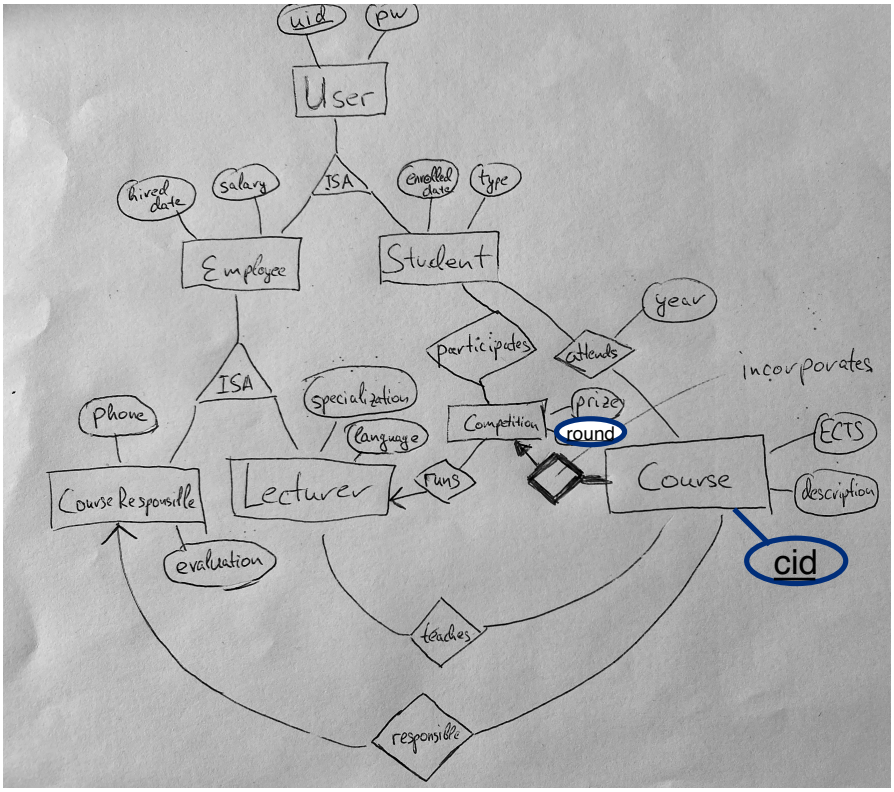
1 Entity–Relationship model (15%)

Kewl University (KU) uses a relational database management system (DBMS) to manage the data about their courses and the students attending them. Their starting point are the following requirements:

- Every course has exactly one course responsible, but may be taught by multiple lecturers.
- Students attend courses (in different years).
- Lecturer and course responsibles are employees of KU.
- Every KU course incorporates at most one competition, which the students can participate in.
- Every competition is run by exactly one lecturer, called the Master of Competition (MC).
- All users (students and employees) authenticate using a login and a password.

1. Which entities and relationships can you identify in the requirements? For each entity set, come up with 2–3 meaningful attributes.
2. Formalize the requirements and your attributes in an E/R diagram. Underline key attributes. Clearly distinguish between uniqueness (➔) and referential integrity (→) constraints. For each arrow briefly argue (1 sentence) why you have used one or the other. Whenever reasonable use weak entities and ISA-hierarchies, but also briefly justify (1 sentence each) their usage.
3. Convert *one* entity set and *one* relationship set of your choice to a relational database schema (write the two corresponding SQL CREATE TABLE commands).

Solution



1. See diagram.
2. See diagram.
3. ```
   CREATE TABLE Lecturer (
     uid VARCHAR PRIMARY KEY,
     password VARCHAR,
     specialization VARCHAR,
     language VARCHAR,
     hireddate DATE,
     salary INT)
   CREATE TABLE Teaches (
     uid VARCHAR,
     cid VARCHAR,
     PRIMARY KEY (uid, cid)
     FOREIGN KEY uid REFERENCES Lecturer,
     FOREIGN KEY cid REFERENCES Course)
   ```

2 Functional dependencies, keys, BCNF (10%)

Consider the following relation schema for reimbursing (W for wage) external lecturers (L for LecturerID) for courses at KU (K for KUcourseID and E for the number of ECTS the course is worth) with the accompanying list of functional dependencies (FDs).

$$\texttt{Payday}(\mathsf{K}, \mathsf{E}, W, \mathsf{L}) \qquad \mathsf{K} \to \mathsf{E} \qquad \mathsf{KL} \to W \qquad \mathsf{EL} \to W$$

1. For each given functional dependency, give a natural language description (1 sentence per FD).
2. Which of the following functional dependencies can be derived? Why/why not (1 sentence for each explanation)?
   (i) $\mathsf{K}W\mathsf{L} \to \mathsf{KE}W$, (ii) $\mathsf{EL} \to \mathsf{K}$, and (iii) $\mathsf{K} \to \mathsf{KE}W\mathsf{L}$
3. List all of `Payday`'s keys. Explain briefly (1 sentence).
4. What are all the superkeys for relation `Payday` that are not keys? Explain briefly (1 sentence).
5. Is relation `Payday` in Boyce-Codd Normal Form (BCNF)? Why/why not? If not, how would you proceed to achieve BCNF? (One sentence per question.)

Solution

1. $\mathsf{K} \to \mathsf{E}$ Courses have an assigned number of ECTS
   $\mathsf{KL} \to W$ Every lecturer gets a fixed wage for a course.
   $\mathsf{EL} \to W$ The wage of the lecturer only depends on the number of ECTS.
2. $\mathsf{K}W\mathsf{L} \to \mathsf{KE}W\mathsf{L} \to \mathsf{KE}W$ easy to derive because $\mathsf{K} \to \mathsf{E}$. $\mathsf{EL} \to \mathsf{K}$ cannot be derived because K is not on a right-hand-side of any dependency. $\mathsf{K} \to \mathsf{KE}W\mathsf{L}$ can not be derived because L is not on a right-hand-side of any dependency.
3. KL is the only key, because KL must be contained in a key and from KL one can obtain everything.
   ```

4. All proper supersets that contain KL: KWL, KEL, KEWL
5. Not BCNF. $K \to E$ violates BCNF, K is not a superkey. Decompose along $K \to E$ to obtain $R1(K, E)$ with $K \to E$ and $R2(K, L, W)$ with $K \to LW$ (both BCNF). (Decomposition along $EL \to K$ also possible, but needs a second iteration.)

3 Relational calculus (RC) (25%)

KU's courses and competitions are organized in four relations with the following schemas:

`Course`	(*courseID* : int,	*title* : string,	*compID* : int,	*capacity* : int,	*examForm* : string)
`Student`	(*studentID* : int,	*name* : string,	*age* : int,	*enrollYear* : int)	
`Attends`	(*studentID* : int,	*courseID* : int,	*year* : int,	*grade* : int)	
`Competes`	(*studentID* : int,	*compID* : int,	*year* : int,	*round* : int,	*points* : int)

Note: The primary key of each relation consists of the underlined attributes. In particular, each course has precisely one competition (identified by *compID*) associated to it. For simplicity you may assume that course titles are unique (but there is still a separation between the title and the course ID).

3.1 Write RC queries

Express the following queries using RC (you are *not* required to use the relational algebra normal form here).

1. What are the names of the students attending the `"DIS"` course in 2022?
2. What are the titles of the courses, which at least one student had to repeat (i.e., attended in two different years)?
3. What are the IDs of the students who have attended a course in some year but did not participate in (any round of) the associated competition in the same year?
4. What are the names of the students who have the top grade (12) in all courses they attended?

Solution

1. $\exists cid, compid, m, e, sid, g, a, ey.$
   $\texttt{Course}(cid, \texttt{"DIS"}, compid, m, e) \wedge \texttt{Attends}(sid, cid, 2022, g) \wedge \texttt{Student}(sid, name, a, ey)$
2. $\exists cid, compid, m, e, sid, y1, y2, g1, g2.$
   $\texttt{Course}(cid, t, compid, m, e) \wedge \texttt{Attends}(sid, cid, y1, g1) \wedge \texttt{Attends}(sid, cid, y2, g2) \wedge \neg y1 = y2$
3. $\exists cid, t, compid, m, e, sid, y, g, a, ey.$
   $\texttt{Course}(cid, t, compid, m, e) \wedge \texttt{Attends}(sid, cid, y, g) \wedge \neg\exists r, p.\ \texttt{Competes}(sid, compid, y, r, p)$
4. $\exists sid, a, ey.\ \texttt{Student}(sid, name, a, ey) \wedge (\forall cid, year, g.\ \texttt{Attends}(sid, cid, year, g) \longrightarrow g = 12)$

3.2 Write RC queries in relational algebra normal form (RANF)

Express the following queries using RC formulas in RANF.

1. What are the IDs of the students who have attended `"MicroB"` but have never attended `"DIS"`.
2. What are the strings that are stored in the database (only student names, course titles, and the exam forms are stored as strings)? **Hint:** the desired query should only have one free variable.

Solution

1. $(\exists \mathtt{cid}, \mathtt{compid}, \mathtt{m}, e, \mathtt{y}, \mathtt{g}.\ \mathtt{Course}(\mathtt{cid}, \texttt{"MicroB"}, \mathtt{compid}, \mathtt{m}, e) \wedge \mathtt{Attends}(\mathtt{sid}, \mathtt{cid}, \mathtt{y}, \mathtt{g})) \wedge \neg$
   $(\exists \mathtt{cid}, \mathtt{compid}, \mathtt{m}, e, \mathtt{y}, \mathtt{g}.\ \mathtt{Course}(\mathtt{cid}, \texttt{"DIS"}, \mathtt{compid}, \mathtt{m}, e) \wedge \mathtt{Attends}(\mathtt{sid}, \mathtt{cid}, \mathtt{y}, \mathtt{g}))$
2. $(\exists \mathtt{cid}, \mathtt{compid}, \mathtt{m}, e.\ \mathtt{Course}(\mathtt{cid}, \mathtt{str}, \mathtt{compid}, \mathtt{m}, e)) \vee$
   $(\exists \mathtt{cid}, \mathtt{t}, \mathtt{compid}, \mathtt{m}.\ \mathtt{Course}(\mathtt{cid}, \mathtt{t}, \mathtt{compid}, \mathtt{m}, \mathtt{str})) \vee$
   $(\exists \mathtt{sid}, \mathtt{a}, ey.\ \mathtt{Student}(\mathtt{sid}, \mathtt{str}, \mathtt{a}, ey))$

3.3 Interpret RC queries

Describe what the following RC queries express in natural language.

1. $\exists cid.\ \mathtt{Attends}(sid, cid, 2022, 12)$
2. $\exists cid, compid, e.\ \mathtt{Course}(cid, t, compid, 100, e) \wedge \neg(e = \texttt{"written"})$
3. $\exists cid, compid, m, e.\ \mathtt{Course}(cid, \texttt{"DIS"}, compid, m, e) \wedge \mathtt{Competes}(sid, compid, 2022, r, p)$

Solution

1. What are the IDs of students who have received at least one 12 in 2022?
2. What are the titles of courses with a capacity of 100 that do not have a written exam.
3. What are the IDs of students competing in a DIS competition in 2022 and what are their points in the individual competition rounds?

4 Relational algebra (RA) (25%)

We consider the same database schema as in the previous exercise.

4.1 RC to RA

Express the three RC queries from Question 3.3 in (extended) RA.

Solution

1. $\pi_{\mathtt{studentID}}(\sigma_{\mathtt{year}=2022 \wedge \mathtt{grade}=12}(\mathtt{Attends}))$
2. $\pi_{\mathtt{title}}(\sigma_{\mathtt{capacity}=100 \wedge \neg \mathtt{exam}=\texttt{"written"}}(\mathtt{Attends}))$
3. $\pi_{\mathtt{studentID},\mathtt{round},\mathtt{points}}(\sigma_{\mathtt{title}=\texttt{"DIS"}}(\mathtt{Course}) \bowtie \sigma_{\mathtt{year}=2022}(\mathtt{Competes}))$

4.2 Write RA queries

Express the following queries using (extended) RA.

1. How many different students attended `"DIS"` since 2020?
2. For every year and every course (ID), what was the average grade of the attending students?
3. For every course (ID), what was the maximum number of students attending (per year)?
4. For every course (ID), what were the years with the most students attending?

Solution

1. $\gamma_{\mathrm{CNT}(\texttt{studentID})}(\delta(\pi_{\texttt{studentID}}(\sigma_{\texttt{title}=\texttt{"DIS"}\wedge\texttt{year}\geq 2020}(\texttt{Course} \bowtie \texttt{Attends}))))$
2. $\gamma_{\texttt{courseID},\texttt{year},\mathrm{AVG}(\texttt{grade})\rightarrow\texttt{avg}}(\texttt{Attends})$
3. $\gamma_{\texttt{courseID},\mathrm{MAX}(\texttt{cnt})\rightarrow\texttt{max}}(\gamma_{\texttt{courseID},\texttt{year},\mathrm{CNT}(\texttt{studentID})\rightarrow\texttt{cnt}}(\texttt{Attends}))$
4. $\texttt{CntAttends}(\texttt{courseID},\texttt{year},\texttt{cnt}) := \gamma_{\texttt{courseID},\texttt{year},\mathrm{CNT}(\texttt{studentID})\rightarrow\texttt{cnt}}(\texttt{Attends});$
   $\pi_{\texttt{courseID},\texttt{year}}(\gamma_{\texttt{courseID},\mathrm{MAX}(\texttt{cnt})\rightarrow\texttt{cnt}}(\texttt{CntAttends}) \bowtie \texttt{CntAttends})$

4.3 Interpret RA queries

Describe what the following RC queries express in natural language.

1. $\pi_{sid1,sid2}(\sigma_{sid1<sid2}(\pi_{studentID\rightarrow sid1,courseID,year}(\texttt{Attends}) \bowtie \pi_{studentID\rightarrow sid2,courseID,year}(\texttt{Attends})))$
2. $\gamma_{\mathrm{CNT}(name)\rightarrow cnt}(\delta(\pi_{name}(\texttt{Student})))$
3. $\pi_{name,title}((\gamma_{\mathrm{MAX}(age)\rightarrow max}\ (\texttt{Student})) \bowtie_{max=age} \texttt{Student}) \bowtie \texttt{Course})$

Solution

1. Which pairs of students (IDs) have attended the same course (in the same year) at least once. Each pair should be listed as many times as they have been attending the same course.
2. How many **different** names of students are there?
3. What are the pairs of names of the oldest students and the existing courses (titles)?

5 SQL (25%)

We consider the same database schema as in the previous two exercises.

5.1 RA to SQL

Express the three RA queries from Question 4.3 in SQL.

Solution

1. ```
   SELECT A.studentID AS sid1, B.studentID AS sid2
   FROM Attends A, Attends B
   WHERE A.courseID = B.courseID AND A.year=B.year AND A.studentID < B.studentID
   ```
2. ```
   SELECT CNT(DISTINCT name) AS cnt
   FROM Student
   ```
3. ```
   SELECT S.name, C.title
   FROM (SELECT MAX(age) AS max) AS M
   JOIN Student S ON M.max = S.age
   CROSS JOIN Course C
   ```

5.2 Write SQL queries

Express the following queries using SQL.

1. What is the grade average of every student (ID) who has attended at least one course?
2. What are the names of students participating in the `"DIS"` competition in 2022 who have not attended this course in 2021 or earlier?
3. Which courses (IDs) were attended by more students in 2022 than is specified as their capacity?
4. Which students (IDs) have won (i.e., gained the highest number of points in) at least one `"DIS"` competition round in 2022?

Solution

1. ```
   SELECT studentID, AVG(grade) -> avg
   FROM Attends
   GROUP BY studentID
   ```
2. ```
   SELECT name
   FROM Student NATURAL JOIN Competes NATURAL JOIN Course
   WHERE year = 2022 AND title = "DIS"
   EXCEPT
   SELECT name
   FROM Student NATURAL JOIN Attends NATURAL JOIN Course
   WHERE year < 2022 AND title = "DIS"
   ```
3. ```
   SELECT courseID
   FROM Course C
   WHERE capacity <
   (SELECT COUNT(DISTINCT studentID)
   FROM Attends A
   WHERE A.courseID = C.courseID AND A.year = 2022)
   ```
4. ```
   SELECT C.studentID FROM
   (SELECT round, MAX(points) -> max
   FROM Course NATURAL JOIN Competes
   WHERE title = "DIS" AND year = 2022
   GROUP BY round) AS M
   ```
   ```

```
    JOIN
    (SELECT studentID, round, points
    FROM Course NATURAL JOIN Competes
    WHERE title = "DIS" AND year = 2022) AS C
    ON M.round = C.round AND M.max = C.points
```

5.3 Modify the database

Express the following modifications to the underlying tables using SQL.

1. Exmatriculate! Delete all students (and the associated course attendance and competition participation data) that enrolled before 2014.
2. Bonus! Increase every student grade by 1.
3. Competition bonus! Increase the grade by 1 for those students who have participated in the associated competition.
4. Create a view `DIS_Attends` for the students attending the latest `"DIS"` course. Note that the meaning of "latest" may change over time as data for the new course years arrives.
5. COVID-19 is over: Change all `"take-home"` exam forms to `"written"`!

Solution

1. ```
   DELETE FROM Attends
   WHERE studentID IN (SELECT studentID FROM Student WHERE enrollYear < 2014)

   DELETE FROM Competes
   WHERE studentID IN (SELECT studentID FROM Student WHERE enrollYear < 2014)

   DELETE FROM Student
   WHERE enrollYear < 2014
   ```
2. ```
   UPDATE Attends
   SET grade = grade + 1
   ```
3. ```
   UPDATE Attends
   SET grade = grade + 1
   WHERE studentID IN (SELECT studentID FROM Course C NATURAL JOIN Competes
                       WHERE C.courseID = courseID)
   ```
4. ```
   CREATE VIEW DIS_Attends AS
   SELECT A.*
   FROM Attends A NATURAL JOIN Course
   WHERE title = "DIS" AND A.year = (SELECT MAX(year)
       FROM Attends NATURAL JOIN Course WHERE title = "DIS")
   ```
5. ```
   UPDATE Course
   SET examForm = "written"
   WHERE examForm = "take-home"
   ```
   ```