Databases Autumn 2025

Hand-In Exercise 1

October 15, 2025

Aiysha Frutiger
Jannick Seper
Luis Tritschler

Total Points	

Task	**Points**

Aiysha Frutiger
Jannick Seper
Luis Tritschler

Task 1

Given excerpt of the relational schema plus added relations and integrety constrain befor the first bullet point.

```
====== Entity ======
CREATE TABLE Lecturer (
        LecturerID  INT PRIMARY KEY,
        FirstName   VARCHAR(255),
        LastName    VARCHAR(255),
        Title       VARCHAR(255)
);

CREATE TABLE Lecture (
        Title               VARCHAR(255) PRIMARY KEY,
        CreditPoints        INT,
        SemesterWeekHours   INT,
        LecturerID          INT,
        FOREIGN KEY (LecturerID) REFERENCES Lecturer (LecturerID)
);

CREATE TABLE Exercise (
        ExID            INT PRIMARY KEY,
        No              INT,
        Semester        VARCHAR(255),
        LectureTitle VARCHAR(255),
        FOREIGN KEY (LectureTitle) REFERENCES Lecture (Title)
);

CREATE TABLE Author (
        AuthorID    INT PRIMARY KEY,
        LastName    VARCHAR(255),
        FirstName   VARCHAR(255),
        Title       VARCHAR(255)
);

CREATE TABLE Task (
        TaskID      INT PRIMARY KEY,
        Points      INT,
        Difficulty  INT,
        Text        VARCHAR(65535),
        AuthorID    INT,
        FOREIGN KEY (AuthorID) REFERENCES Author (AuthorID)
);

====== Relationships ======
```

```
CREATE TABLE Contains (
        ExID      INT,
        TaskID    INT,
        PRIMARY KEY (ExID, TaskID),
        FOREIGN KEY (ExID)  REFERENCES Exercise (ExID),
        FOREIGN KEY (TaskID) REFERENCES Task (TaskID),
        Sequence  INT
);

CREATE TABLE Consists_of (
        SuperTaskID INT,
        SubTaskID   INT,
        PRIMARY KEY (SuperTaskID, SubTaskID),
        FOREIGN KEY (SuperTaskID) REFERENCES Task (TaskID),
        FOREIGN KEY (SubTaskID)   REFERENCES Task (TaskID),
        Sequence    INT
);

====== Integrity ======
CREATE ASSERTION contains_only_super_tasks
  CHECK ( NOT EXISTS (
    SELECT *
    FROM Contains c
    WHERE EXISTS (
      SELECT *
      FROM Consists_of co
      WHERE co.SubTaskID = c.TaskID
    )
  ) );

CREATE ASSERTION consists_of_non_recursive
  CHECK ( NOT EXISTS (
    SELECT *
    FROM Consists_of x
    WHERE EXISTS (
      SELECT *
      FROM Consists_of y
      WHERE y.SuperTaskID = x.SubTaskID
    )
  ) );
```

- *The title of the lecture has to be unique and may not be altered if any exercise is available for the lecture.*
  The first part of the point is already enforced bc the title of the lecture is a `PRIMARY KEY` and therefore must be unique. The second part can be enforced with this addition.

```
CREATE TABLE Exercise (
        ExID            INT PRIMARY KEY,
        No              INT,
        Semester        VARCHAR(255),
        LectureTitle    VARCHAR(255),
        FOREIGN KEY (LectureTitle) REFERENCES Lecture (Title)
->      ON UPDATE RESTRICT
);
```

- *For a lecture, no more than 10 credit points may be awarded.*
  This part can be enforced with this assertion.

```
CREATE ASSERTION no_more_than_10_credits
        CHECK (NOT EXISTS (
                SELECT * FROM Lecture
                WHERE CreditPoints > 10))
        NOT DEFERRABLE;
```

- *Lecturers may give multiple lectures.*
  This part is beeing enforced with this.

```
CREATE TABLE Lecture (
        Title               VARCHAR(255) PRIMARY KEY,
        CreditPoints        INTEGER,
        SemesterWeekHours   INTEGER,
        LecturerID          INTEGER,
->      FOREIGN KEY (LecturerID) REFERENCES Lecturer (LecturerID)
);
```

- *A lecture may include several exercises. An exercise always belongs to exactly one lecture.*
  The first part is already enforced via the foreign key while for the second part we have to add the `NOT NULL` so we guaratee that e exersise must be in one lecture.

```
CREATE TABLE Exercise (
        ExID            INT PRIMARY KEY,
        No              INT,
        Semester        VARCHAR(255),
->      LectureTitle    VARCHAR(255) NOT NULL,
->      FOREIGN KEY (LectureTitle) REFERENCES Lecture (Title)
        ON UPDATE RESTRICT
);
```

- *Before a new author is entered into the system, it should be checked that no other author with the same first name, last name and title is present.*
  This part is beeing enforced with this.

```
CREATE TABLE Author (
        AuthorID    INT PRIMARY KEY,
        LastName    VARCHAR(255),
        FirstName   VARCHAR(255),
        Title       VARCHAR(255),
->      UNIQUE (FirstName, LastName, Title)
);
```

Task 2

Task 3

Task 4

NOT sure yeeeet.