

Databases Autumn 2025

Hand-In Exercise 1

October 15, 2025

Aiysha Frutiger
Jannick Seper
Luis Tritschler

Total Points	
---------------------	--

Task 1

Given excerpt of the relational schema plus added relations and integrity constrain before the first bullet point.

```
1  ===== Entity =====
2  CREATE TABLE Lecturer (
3      LecturerID INT PRIMARY KEY,
4      FirstName  VARCHAR(255),
5      LastName   VARCHAR(255),
6      Title      VARCHAR(255)
7  );
8
9  CREATE TABLE Lecture (
10     Title          VARCHAR(255) PRIMARY KEY,
11     CreditPoints   INT,
12     SemesterWeekHours INT,
13     LecturerID     INT,
14     FOREIGN KEY (LecturerID) REFERENCES Lecturer (LecturerID)
15 );
16
17 CREATE TABLE Exercise (
18     ExID          INT PRIMARY KEY,
19     No            INT,
20     Semester      VARCHAR(255),
21     LectureTitle  VARCHAR(255),
22     FOREIGN KEY (LectureTitle) REFERENCES Lecture (Title)
23 );
24
25 CREATE TABLE Author (
26     AuthorID     INT PRIMARY KEY,
27     LastName      VARCHAR(255),
28     FirstName     VARCHAR(255),
29     Title         VARCHAR(255)
30 );
31
32 CREATE TABLE Task (
33     TaskID        INT PRIMARY KEY,
34     Points        INT,
35     Difficulty    INT,
36     Text          VARCHAR(65535),
37     AuthorID      INT,
38     FOREIGN KEY (AuthorID) REFERENCES Author (AuthorID)
39 );
40
41 ===== Relationships =====
```

```
42 CREATE TABLE Contains (
43     ExID      INT,
44     TaskID    INT,
45     PRIMARY KEY (ExID, TaskID),
46     FOREIGN KEY (ExID) REFERENCES Exercise (ExID),
47     FOREIGN KEY (TaskID) REFERENCES Task (TaskID),
48     Sequence  INT
49 );
50
51 CREATE TABLE Consists_of (
52     SuperTaskID INT,
53     SubTaskID   INT,
54     PRIMARY KEY (SuperTaskID, SubTaskID),
55     FOREIGN KEY (SuperTaskID) REFERENCES Task (TaskID),
56     FOREIGN KEY (SubTaskID)   REFERENCES Task (TaskID),
57     Sequence      INT
58 );
59
60 ===== Integrity =====
61 CREATE ASSERTION contains_only_super_tasks
62 CHECK ( NOT EXISTS (
63     SELECT *
64     FROM Contains c
65     WHERE EXISTS (
66         SELECT *
67         FROM Consists_of co
68         WHERE co.SubTaskID = c.TaskID
69     )
70 ) );
71
72 CREATE ASSERTION consists_of_non_recursive
73 CHECK ( NOT EXISTS (
74     SELECT *
75     FROM Consists_of x
76     WHERE EXISTS (
77         SELECT *
78         FROM Consists_of y
79         WHERE y.SuperTaskID = x.SubTaskID
80     )
81 ) );
```

- *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.

```
1 CREATE TABLE Exercise (
2     ExID          INT PRIMARY KEY,
3     No            INT,
4     Semester      VARCHAR(255),
5     LectureTitle  VARCHAR(255),
6     FOREIGN KEY (LectureTitle) REFERENCES Lecture (Title)
7     ON UPDATE RESTRICT
8 );
```

- *For a lecture, no more than 10 credit points may be awarded.*

This part can be enforced with this assertion.

```
1 CREATE ASSERTION no_more_than_10_credits
2     CHECK (NOT EXISTS (
3         SELECT * FROM Lecture
4         WHERE CreditPoints > 10))
5     NOT DEFERRABLE;
```

- *Lecturers may give multiple lectures.*

This part is beeing enforced with this.

```
1 CREATE TABLE Lecture (
2     Title          VARCHAR(255) PRIMARY KEY,
3     CreditPoints   INTEGER,
4     SemesterWeekHours INTEGER,
5     LecturerID     INTEGER,
6     FOREIGN KEY (LecturerID) REFERENCES Lecturer (LecturerID)
7 );
```

- *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 exercise must be in one lecture.

```
1 CREATE TABLE Exercise (
2     ExID          INT PRIMARY KEY,
3     No            INT,
4     Semester      VARCHAR(255),
5     LectureTitle  VARCHAR(255) NOT NULL,
6     FOREIGN KEY (LectureTitle) REFERENCES Lecture (Title)
7     ON UPDATE RESTRICT
8 );
```

- *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 being enforced with this.

```
1 CREATE TABLE Author (  
2     AuthorID    INT PRIMARY KEY,  
3     LastName    VARCHAR(255),  
4     FirstName   VARCHAR(255),  
5     Title       VARCHAR(255),  
6     → UNIQUE (FirstName, LastName, Title)  
7 );
```

Task 2

Task 3

Task 4

When we say something is **DEFERRABLE INITIALLY DEFERRED**, it means that the database will wait until the end of a transaction before checking certain rules, such as foreign key constraints. This is useful when two tables depend on each other. For example, a **Lecture** table and a **Lecturer** table. A lecture record might point to the lecturer who gives it, while the lecturer record might also depend on the lecture. If we try to insert both of these in the same transaction, the database might raise an error because when we insert the first one, the other does not yet exist. If the rule is **NOT DEFERRABLE**, the database checks immediately, and the operation fails. If it is **DEFERRABLE INITIALLY IMMEDIATE**, we would need to manually tell the database to delay the check. However, if it is **DEFERRABLE INITIALLY DEFERRED**, the database automatically waits until the transaction ends to perform the checks. This allows both inserts to complete first, and the database only verifies the rules afterward, which helps prevent errors.