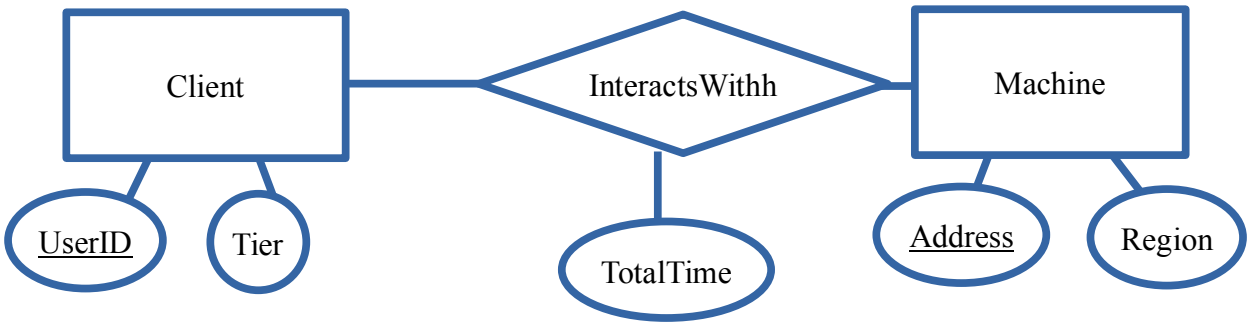
Section 8 Worksheet

ER → SQL

Coming up with ER diagrams is relatively simple compared to coming up with a representation in SQL. In this part we will try to convert some ER diagrams into SQL CREATE TABLE statements. **For the following ER Diagrams and respective specifications, come up with the appropriate schema and implement it in SQL.** Feel free to abbreviate names to save time and assume attribute types as appropriate.



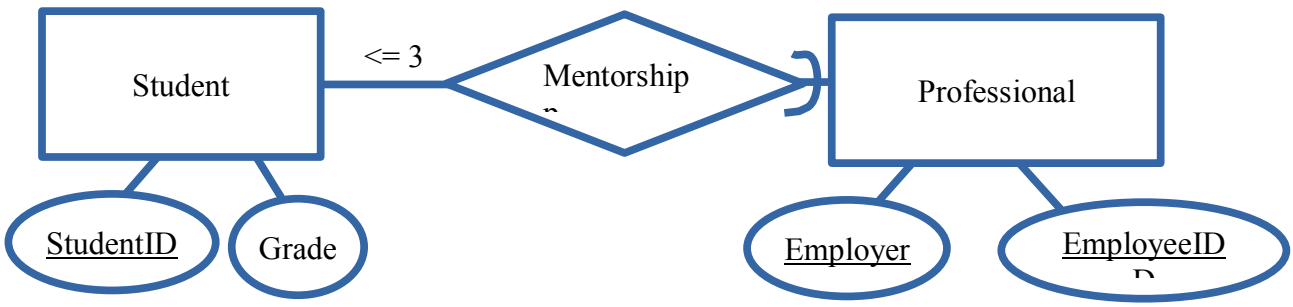
Other Specification:

If a Client is removed, we want to retain the interaction history.

If a Machine is removed, we don't want to retain the interaction history.

Any updates for a Client or Machine should also update respective interaction records.

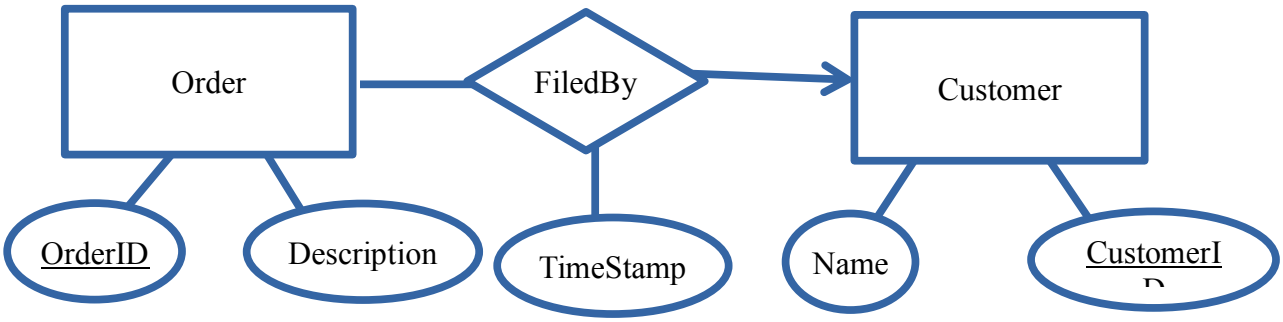
```
CREATE TABLE Client (
   UserID INT PRIMARY KEY,
   Tier VARCHAR(20)
);
CREATE TABLE Machine (
   Address VARCHAR(40) PRIMARY KEY,
   Region VARCHAR(50)
);
CREATE TABLE InteractsWith (
   TotalTime FLOAT,
   UserID INT REFERENCES Client
      ON UPDATE CASCADE
      ON DELETE SET NULL,
   Address VARCHAR(40) REFERENCES Machine
      ON UPDATE CASCADE
      ON DELETE CASCADE,
   PRIMARY KEY (UserID, Address)
);
```



Other Specification:
Only Students that are grade 8 or higher are allowed to have mentorships.
Any updates for a Student or Professional should also update respective interaction records.

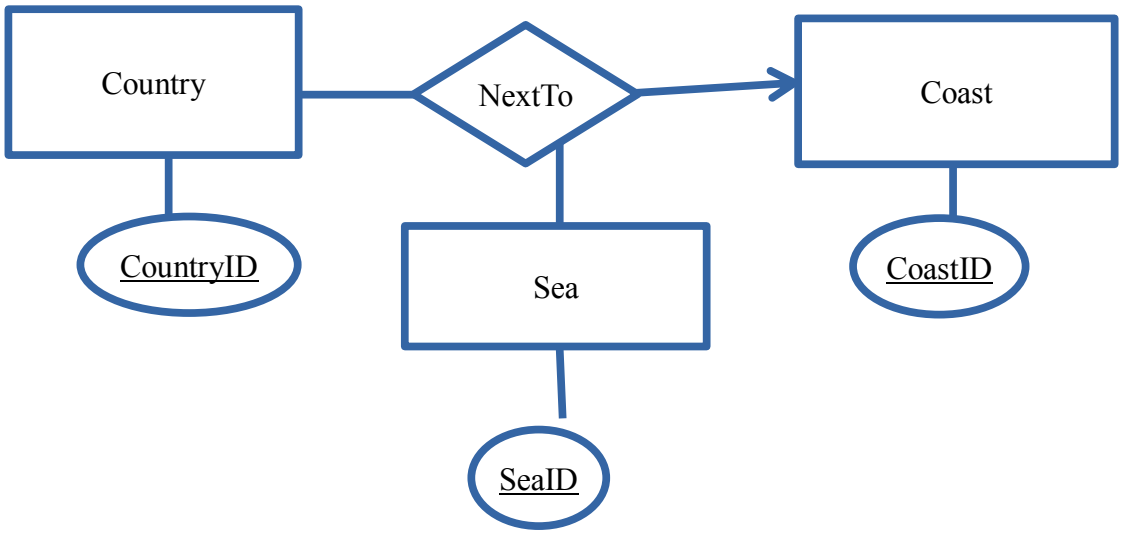
```
CREATE TABLE Student (
   StudentID VARCHAR(50) PRIMARY KEY,
   Grade INT CHECK (Grade >= 8),
   MentorEmployer VARCHAR(100) NOT NULL,
   MentorID VARCHAR(50) NOT NULL,
   FOREIGN KEY (MentorEmployer, MentorID) REFERENCES Professional(Employer, EmployeeID)
      ON UPDATE CASCADE
);
CREATE TABLE Professional (
   Employer VARCHAR(100),
   EmployeeID VARCHAR(50),
   PRIMARY KEY (Employer, EmployeeID)
);
CREATE ASSERTION MyAssert CHECK (
  3 >= ALL (SELECT count(*)
              FROM Student AS S
             GROUP BY S.MentorEmployer, S.MentorID)
);

-- alternatively you could use a trigger
CREATE TRIGGER MyTrigger
ON Student
AFTER INSERT
AS IF (3 < ANY (SELECT count(*)
                  FROM Student AS S
                 GROUP BY S.MentorEmployer, S.MentorID))
   ROLLBACK TRANSACTION;
```



Other Specification:
If a customer is removed we should have all respective orders altered so they are from the default anonymous user with CustomerID 0.
Order descriptions should be sufficiently specific, so we need to make sure the description is at least 25 characters long.
Any updates for an Order or Customer should also update respective interaction records.

```
CREATE TABLE Order (
   OrderID INT PRIMARY KEY,
   Description VARCHAR(255) CHECK (len(Description) >= 25),
   TimeStamp VARCHAR(50),
   CustomerID INT REFERENCES Customer
      ON UPDATE CASCADE
      ON DELETE SET DEFAULT
);
CREATE TABLE Customer (
   Name VARCHAR(100),
   CustomerID INT DEFAULT 0 PRIMARY KEY,
);
```



Other Specification:
Any updates for an Country, Sea, or Coast should also update respective interaction records.

```
CREATE TABLE Country (
   CountryID INT PRIMARY KEY
);
CREATE TABLE Sea (
   SeaID INT PRIMARY KEY
);
CREATE TABLE Coast (
   CoastID INT PRIMARY KEY
);
CREATE TABLE NextTo (
   CountryID INT REFERENCES Country
      ON UPDATE CASCADE,
   SeaID INT REFERENCES Sea
      ON UPDATE CASCADE,
   CoastID INT REFERENCES Coast
      ON UPDATE CASCADE,
   PRIMARY KEY (CountryID, SeaID)
);
```

1. (**14AU Final Q10**)
   A lossless decomposition is one in which all of the data in the original table is recovered exactly, if the decomposed tables are combined with natural joins.

   Every BCNF decomposition is lossless.
   TRUE or FALSE

2. (**17WI Final Q1**)
   BCNF is a lossless decomposition and it does not preserve all functional dependencies.
   TRUE or FALSE

3. (**14WI Final Q2)**
   Consider Relation R(ABCD). and functional dependencies (FDs): BD → AC; AB → D; AC → B.

   This relation is in Boyce-Codd Normal Form (BCNF).
   TRUE or FALSE.

   Identify a key (not a superkey)
   BD or AB or AC

4. (**14 WI Final Q2)**
   Consider Relation R(ABCDE). and functional dependencies (FDs): A → C; B → AE; E → D.

   This relation is in Boyce-Codd Normal Form (BCNF).
   TRUE or **FALSE**

   Identify a key (not a superkey)
   B

5. (**15AU Final Q2c**)
   Consider the following relational schema and set of functional dependencies.
   R(A,B,C,D,E,F,G) with functional dependencies: E → C G → AD B → E C → BF

   Give one example of non-trivial functional dependency implied by the ones above:
   Many solutions are possible. One example is E → BF.

   Compute E +, the closure of E.
   {E} + = {B, C, E, F}

(17WI Final Q4)
Given R(A, B, C, D, E), and functional dependencies: A → C, BD → A, D → E

a) Decompose R into BCNF. In each step, explain which functional dependency you used to decompose and explain why further decomposition is needed. Your answer should consist of a list of table names and attributes. Make sure you indicate the keys for each relation.

One possible decomposition:

1. Use A → C:

Decompose R into R1(A, C) and T(A, B, D, E)

T violates BD → A and D → E, so we need to further decompose T

2. Use D → E:

Decompose T into R2(B, D, A) and R3(D, E)

Final relations: R1(<u>A</u>, C), R2(<u>B, D</u>, A), and R3(<u>D</u>, E)

Given R(A, B, C, D, E), and functional dependencies: A → C, BD → A, D → E

b) i) Suppose we decomposed R into R_1(B, C, D), R_2(A, B, D), and R_3(A, E) (this is not the answer to a) by the way...). Is this decomposition lossless? Verify this by showing the final tableau after applying the chase algorithm. Make sure you indicate which functional dependency (FD) did each row in the tableau come from.

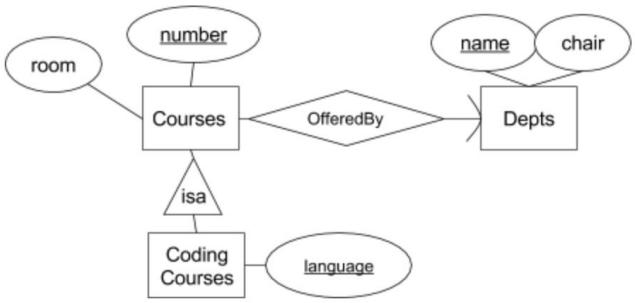
A	B	C	D	E	Which FD?
a1	b	c	d	e1	(initial tableau)
a	b	c1	d	e2	(initial tableau)
a	b1	c2	d1	e	(initial tableau)
a	b	c	d	e1	Apply BD → A
a	b	c1	d	e2	Apply BD → A
a	b1	c2	d1	e	Apply BD → A
a	b	c	d	e1	Apply A → C
a	b	c	d	e2	Apply A → C
a	b1	c	d1	e	Apply A → C

Final tableau shown in blue (D→ E does not induce any further changes)

ii) Decomposition is: Lossy Lossless

c) Convert the E/R diagram below to relations in BCNF form. Assume no values are `NULL`, and the arrow between `OfferedBy` and `Depts` is a round one. Include all keys and foreign keys. Use the following notation and explicitly state foreign key relationships. For instance:

R(a, b)
S(c, d) -- c is a foreign key to R



Courses(number, room, name) -- name is foreign key to Depts

CodingCourses(language, number) -- number is foreign key to Courses

Depts(name, chair)

Language should not be a key of CodingCourses in the diagram.

This solution is copied from the 17WI Final, but it is actually invalid to have a key in a subclass, so the correct answer should be CodingCourses(language, number), where number is still a foreign key.