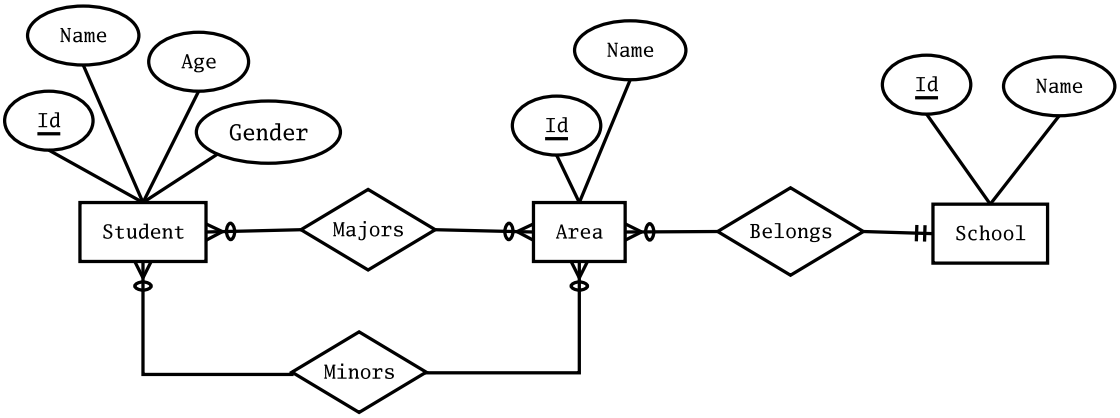
1 The Schema

This exercise uses the example of students, who may major or minor in zero or more areas, where each area belongs to one school. The ER diagram is shown in figure 1.



Notice this is a simplified example. We are using the student's age, rather than their date of birth etc.

Figure 1: ER Diagram for exercise

The SQL Schema with sample data is as follows:

1.1 School

```
1 CREATE TABLE School (
2   Id    INTEGER PRIMARY KEY,
3   Name  VARCHAR(45) UNIQUE NOT NULL
4 );
```

id	name
1	Computing and Software Engineering
2	Basket Weaving and Juggling

1.2 Area

```
1 CREATE TABLE Area (
2   Id     INTEGER PRIMARY KEY,
3   Name   VARCHAR(25) UNIQUE NOT NULL,
4   School INTEGER NOT NULL
5      REFERENCES School(id)
6 );
```

id	name	school
1	Computer Science	1
2	Software Engineering	1
3	Information Technology	1
4	Basket Weaving	2
5	Juggling	2

1.3 Student

```
1 CREATE TABLE Student (
2   Id     INTEGER PRIMARY KEY,
3   Name   VARCHAR(30) NOT NULL,
4   Age    INTEGER NOT NULL,
5   Gender CHAR(1) NOT NULL
6     CHECK(Gender IN ('M','F'))
7 );
```

id	name	age	gender
1	Orlando Karam	30	M
2	Lina Colli	29	F
3	John Smith	18	M
4	Jane Smith	19	F
5	Ruben Rada	20	M
6	Manuel Morales	21	M

1.4 Majors

```
CREATE TABLE Majors (
  Student INTEGER
      REFERENCES Student(Id),
  Area    INTEGER REFERENCES Area(Id),
  PRIMARY KEY(Student,Area)
);
```

student	area
1	4
2	1
3	2
4	3
4	4

1.5 Minors

```
CREATE TABLE Minors (
  Student INTEGER
      REFERENCES Student(Id),
  Area    INTEGER REFERENCES Area(Id),
  PRIMARY KEY(Student,Area)
);
```

student	area
1	5
2	1
2	4
3	5
4	5

2 SQL Examples

2.1 Show the id of the students who are majoring in the area with id 1

For this one, we just need the information in the Majors table, so the following query works:

```
SELECT Student
FROM Majors
WHERE Area=1
```

2.2 Show the id of the students who are majoring in either the area with id 1 or the area with id 2

Now you try ...

```
SELECT

FROM

WHERE
```

2.3 Show the id of the students who are majoring in BOTH the area with id 1 or the area with id 2

Now you try ...

```
SELECT

FROM

WHERE
```

for this one we need IN, or at least the simple joins won't work

2.4 Show the *name* of the students who are majoring in the area with id 1

For this one, we need information from both the Student table and the Majors table. We can do this query with a join statement, as follows:

```
1 SELECT S.Name
2 FROM Student S JOIN Majors M ON (M.Student=S.Id)
3 WHERE M.Area =1
```

This uses an explicit join. Can you do it implicitly, that is, without using the JOIN keyword ?

We can also do this using a subquery and the IN predicate, as follows:

```
1 SELECT Name
2 FROM Student
3 WHERE Id IN (
4    SELECT Student
5    FROM Majors
6    WHERE Area=1
7 )
```

A more esoteric SQL feature allows us to append the keyword ALL or ANY to a comparison operator, and use it to compare one element to all elements in a list; =ANY is basically equivalent to IN.

Or using EXISTS (notice this is a correlated subquery, since the subquery references the outside query; a naive execution of this query would require the subquery to be re-executed for each row of the outside query):

```
1 SELECT Name
2 FROM Student S
3 WHERE EXISTS (
4    SELECT *
5    FROM Majors M
6    WHERE M.Area=1 AND M.Student=S.Id
7 )
```

Which one is easier for you to understand ? IN or EXISTS ?

2.5 Show the ids of the students who are majoring in the area *named* 'Computer Science'

For doing this one with a join, we need to join Area with Majors (so we can get both the student id and the area's name).

Now you try ...

```
SELECT M.Student
FROM Majors M JOIN Area A ON (                          )
WHERE
```

We can also do this with an IN

Now you try ...

```
SELECT Student
FROM Majors M WHERE              IN (
   SELECT
   FROM Area
   WHERE
)
```

2.6 Show the name of the students who are majoring in the area with *name* 'Computer Science'

For this one, we need to join three tables. Notice the syntax for the join; basically a whole clause, A JOIN B ON (...) syntactically becomes equivalent to just a table name;

```
1 SELECT S.Name
2 FROM Student S JOIN Majors M ON (M.Student=S.Id)
3    JOIN Area A ON (M.Area=A.Id)
4 WHERE A.Name='Computer Science'
```

Can you do this one with implicit joins ? How about using only subqueries ? A combination of joins and subqueries ?

2.7 Show the id of the students who are majoring in any area within the *school* with id 1

Now, how is this one different from the last one? Can we get our info from the same tables ?

Now you try ...

```
SELECT
FROM         JOIN          ON (                    )
WHERE
```

2.8 Show the *name* of the students who are majoring in any area within the school with id 1

Now, which other table do we need ?

Now you try ...

```
SELECT
FROM         JOIN          ON (                    )
      JOIN              ON (                    )
WHERE
```

2.9 Show the name of the students who are majoring in any area within the *school* named 'Computing and Software Engineering'

What about this one ?

Now you try ...

```
SELECT
FROM         JOIN          ON (                    )
      JOIN              ON (                    )
      JOIN              ON (                    )
WHERE
```

2.10 Show the id of each area, with the number of students majoring in that area

For this one we need to do an aggregate (COUNT), and a GROUP BY clause.

```
1 SELECT Area, COUNT(*) AS NumMajors
2 FROM Majors
3 GROUP BY Area
```

This is almost right; what happens if an area has no majors ? we discuss that issue in the next item

2.11 Show the *Name* of each area, with the number of students majoring in that area

For this one, we need to do a join and then count. Now, shall we do an outer join ? Why or why not ?

Complete the query

```
SELECT
FROM                              JOIN          ON (                    )
GROUP BY
```

For this problem, we probably ought to use an OUTER join; if not, areas that have no majors, will NOT appear in the result. So, we need to use an OUTER join, making sure all rows in Area appear (so it's left or right depending on whether Area is to the left or right of the JOIN keyword).

We also need to modify our COUNT expression. If we do COUNT(*) we count all rows; if an area has no majors, there would still be a row in the outer join, so COUNT(*) would yield 1 !. If we include a field (or expression) inside the COUNT, rows where that field is NULL will NOT be counted; so you could use any field of the Majors table; I tend to use the field referenced in the join condition (Majors.Area in this case).

So, my solution would look like this:

```
SELECT A.Id, A.Name, COUNT(M.Area)
FROM Area A LEFT OUTER JOIN Majors M ON (M.Area=A.Id)
GROUP BY A.Id, A.Name
```

2.12 Show the id of each area, with the number of students majoring in that area. Make sure ALL areas appear, even those who have no students majoring on it

Now, this is similar to the two last items. In fact, this is (arguably) the same as the one two items above; we are just clarifying what happens to areas with no majors. With the requirement that those areas appear with a count of 0, our query needs to be like the previous one. We need to join majors with areas.

```
SELECT A.Id, COUNT(M.Area)
FROM Area A LEFT OUTER JOIN Majors M ON (M.Area=A.Id)
GROUP BY A.Id
```

2.13 Show the *name* of each *school*, with the number of students majoring in that school. Make sure ALL schools appear, even those who have no students majoring on it

For this one, we need to join school with area and major. Which ones need to be outer joins ?

You try ...

```
SELECT
FROM                              JOIN          ON (                        )
                          JOIN             ON (                   )
GROUP BY
```

Now, the only problem with a query like this is that if a student is majoring in two areas within the same school that student would be counted twice. We can solve that by adding a DISTINCT keyword *inside* the COUNT, to count only unique student ids (again, NULLs are not counted) as follows:

```
SELECT S.Name, COUNT(DISTINCT M.Student)
FROM Area A JOIN Majors M ON ( M.Area=A.Id )
         RIGHT OUTER JOIN School S ON (A.School=S.Id)
GROUP BY S.Id, S.Name
```

Another possibility is to first get the list of school and students (with a SELECT DISTINCT) and then do the counting. In SQL we can also put a subquery in the FROM clause, instead of just the name of a table, as follows:

```
SELECT Name, COUNT(Student)
FROM (
   SELECT Distinct S.Id, S.Name, M.Student
   FROM Area A JOIN Majors M ON ( M.Area=A.Id )
         RIGHT OUTER JOIN School S ON (A.School=S.Id)
) Temp
GROUP BY Id, Name
```

Notice we are aliasing this subquery with the name temp; PostgreSQL requires us to add that alias; Oracle accepts it but doesn't require it.

2.14 Show the Names of all students who are majoring in EITHER area 3 or area 4

```
SELECT S.Name
FROM Student S JOIN Majors M ON (M.Student=S.Id)
WHERE M.Area=3 OR M.Area=4
```

We can also do
WHERE Area IN (3,4)

2.15 Show the Names of all students who are majoring in BOTH area 3 and area 4

Here a common mistake would be to do something like this:

WRONG solution

```
SELECT S.Name
FROM Student S JOIN Majors M ON (M.Student=S.Id)
WHERE M.Area =3 AND M.Area=4
```

notice this query would NOT work, since the two predicates in the WHERE clause refer to THE SAME row in M. Since the same row in M CANNOT have two values for the same attribute, this query would return an empty set.

We can solve this problem with a JOIN, but we need to join TWICE with the Majors table. We do need to change the alias for the table, to be able to refer to rows in both instances of the Majors table. The query would look as follows:

```
SELECT S.Name
FROM Student S JOIN Majors M1 ON (M1.Student=S.Id)
   JOIN Majors M2 ON (M2.Student=S.Id)
WHERE M1.Area =3 AND M2.Area=4
```

Although joining twice with the same table shouldn't scare you, the query looks much better using subqueries and the IN predicate, as follows:

```
SELECT Name
FROM Student
WHERE Id IN (
   SELECT Student
   FROM Majors     WHERE Area=3
)
AND Id IN (
   SELECT Student
   FROM Majors     WHERE Area=4
)
```

Just to illustrate, here is another solution using INTERSECTS inside a subquery:

```
SELECT Name
FROM Student
WHERE Id IN (
      SELECT Student
      FROM Majors
      WHERE Area=3
   INTERSECT
      SELECT Student
      FROM Majors
      WHERE Area=4
)
```

2.16 Show the Ids of all students who are NOT majoring in area 3

Queries that involve negation tend to confuse many people, so we will discuss this query in depth. Try to NOT look at the solutions, but think it through.

First, using the sample data, calculate the results. Which student id's should appear ? Now, try to write a SQL query for it.

Many people would write a query like this:

WRONG

```
1 SELECT Student
2 FROM Majors
3 WHERE Area!=3
```

This is a completely different statement. This will show the students who are majoring in another are, different from 3, whether they are also majoring in area 3 or not. This would include students who are majoring in area 3 but also in another area (student 4 for our sample data), and will fail to include students who are not majoring in any area.

Another solution that is not quite right would be like this:

WRONG

```
1 SELECT Student
2 FROM Majors
3 WHERE Student NOT IN (
4    SELECT Student
5    FROM Majors
6    WHERE Area=3
7 )
```

How can you avoid duplicate rows appearing in the solution ? Also, try the query with area 1, which student appears twice ?

This one might produce duplicate rows (not with our sample data, but you can change the condition to those not in area 1 and see what happens) and also fail to include those who have not declared a major.

One right solution is like this:

```
1 SELECT Id
2 FROM Student
3 WHERE Id NOT IN (
4    SELECT Student
5    FROM Majors
6    WHERE Area=3
7 )
```

Can you write this solution in another way ? How about using EXISTS ?

3 More Exercises

Provide SQL queries to show:

1. The names of all students who are majoring in area 1 but not minoring in area 3
2. The names of all students who are majoring in area 1 and minoring in area 3
3. The average age of all students who are majoring in area 1
4. The name of each area with the number of students who are *minoring* on it, plus the age of the oldest student majoring on it, and the average age of all students majoring on it.
5. The name of each school, with the number of areas that belong to it
6. The name of each school, with the average age of their majors
7. The name of each area, with both the number of students majoring on it and the number of students minoring on it. Areas with no students appear with a count of 0
8. The name of each school, with the average number of majors per area
9. The name of the oldest student
10. The name of the oldest student majoring in area 1