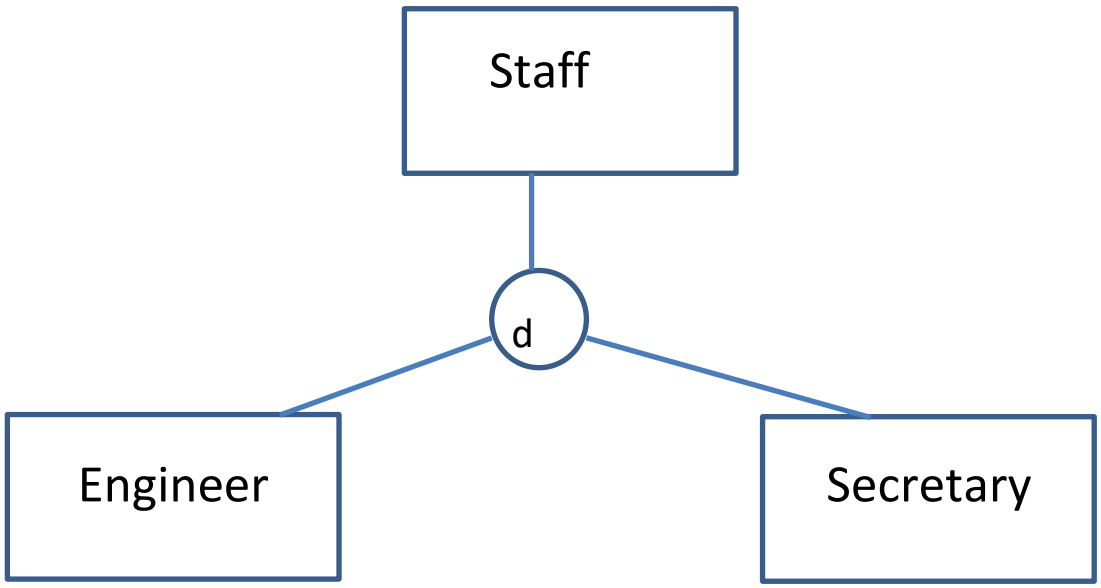
Superclasses Exercise

Disjoint Subclasses

We want to implement this scenario




```
CREATE TABLE super_staff
(
staff_id          VARCHAR(5),
staff_name        VARCHAR(10),
age               NUMBER(3),
salary            NUMBER(6,2),
PRIMARY KEY (staff_id)
);

CREATE TABLE sub_engineer
(
staff_id          VARCHAR(5),
specialism        VARCHAR(10),
PRIMARY KEY (staff_id),
FOREIGN KEY (staff_id) REFERENCES super_staff(staff_id)
);

CREATE TABLE sub_secretary
(
staff_id          VARCHAR(5),
wordspm           NUMBER(3),
PartFull          VARCHAR(1),
PRIMARY KEY (staff_id),
FOREIGN KEY (staff_id) REFERENCES super_staff(staff_id)
);
```

```
INSERT INTO super_staff VALUES ('I0001', 'Wilson', 50, 12.99);
INSERT INTO super_staff VALUES ('I0002', 'Jones', 22, 9.99);
INSERT INTO super_staff VALUES ('I0003', 'Brown', 45, 7.99);

INSERT INTO super_staff VALUES ('I0004', 'Smith', 73, 9.99);
INSERT INTO super_staff VALUES ('I0005', 'Freeman', 22, 10.00);
INSERT INTO super_staff VALUES ('I0006', 'Moore', 44, 15.00);

INSERT INTO sub_engineer VALUES ('I0001', 'Metals');
INSERT INTO sub_engineer VALUES ('I0002', 'Wood');
INSERT INTO sub_engineer VALUES ('I0003', 'Plastics');

INSERT INTO sub_secretary VALUES ('I0004', 124, 'F');
INSERT INTO sub_secretary VALUES ('I0005', 200, 'F');
INSERT INTO sub_secretary VALUES ('I0006', 155, 'P');
```

Note how there is no overlap between the members of staff in the subclasses.

Write a query to list all fields of the engineer staff members.

```
SELECT super_staff.staff_id, staff_name, age, salary, specialism
FROM super_staff, sub_engineer
WHERE super_staff.staff_id = sub_engineer.staff_id;
```

Write a query to list all fields of the secretary staff members.

```
SELECT super_staff.staff_id, staff_name, age, salary, wordspm, partfull
FROM super_staff, sub_secretary
WHERE super_staff.staff_id = sub_secretary.staff_id;
```

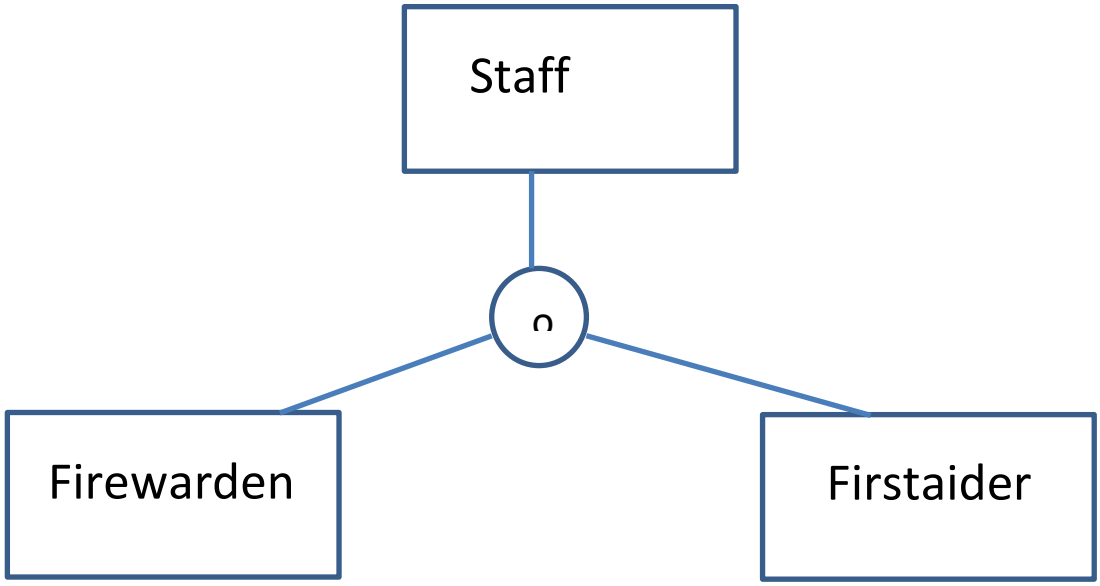
Write a query to list all members of staff, but only the common fields. What if there were more staff members who were not engineers or secretaries?

```
SELECT super_staff.staff_id, staff_name, age, salary
FROM super_staff, sub_engineer
WHERE super_staff.staff_id = sub_engineer.staff_id;
UNION
SELECT super_staff.staff_id, staff_name, age, salary
FROM super_staff, sub_secretary
WHERE super_staff.staff_id = sub_secretary.staff_id;

-- You could then create a view based on this to make other queries easier

CREATE VIEW all_staff AS
SELECT super_staff.staff_id, staff_name, age, salary
FROM super_staff, sub_engineer
WHERE super_staff.staff_id = sub_engineer.staff_id;
UNION
SELECT super_staff.staff_id, staff_name, age, salary
FROM super_staff, sub_secretary
WHERE super_staff.staff_id = sub_secretary.staff_id;
```

Overlapping Subclasses

The above example works because these are disjoint subclasses. A member of staff can either be an engineer or a secretary but not both.

What if they are overlapping subclasses - where a member of staff can be a member of more than one subclass - such as if there were a number of roles that a person could take on, each of which has its own associated data values.



(Super) Staff table and data as before

```
CREATE TABLE super_staff
(
staff_id    VARCHAR(5),
staff_name  VARCHAR(10),
age         NUMBER(3),
salary      NUMBER(6,2),
PRIMARY KEY (staff_id)
);

CREATE TABLE sub_firstaider
(
staff_id    VARCHAR(5),
skill_level VARCHAR(10),
bonus       NUMBER(4,2),
PRIMARY KEY (staff_id),
FOREIGN KEY (staff_id) REFERENCES super_staff(staff_id)
);

CREATE TABLE sub_firewarden
```

```
(
staff_id    VARCHAR(5),
phone_num    NUMBER(10),
bonus       NUMBER(4,2),
PRIMARY KEY (staff_id),
FOREIGN KEY (staff_id) REFERENCES super_staff(staff_id)
);

INSERT INTO sub_firstaider VALUES ('I0001', 'Gold', 23.4);
INSERT INTO sub_firstaider VALUES ('I0002', 'Silver', 44.8);
INSERT INTO sub_firstaider VALUES ('I0003', 'Bronze', 72.2);
INSERT INTO sub_firstaider VALUES ('I0004', 'Gold', 23.4);
INSERT INTO sub_firstaider VALUES ('I0005', 'Silver', 44.8);
INSERT INTO sub_firstaider VALUES ('I0006', 'Bronze', 72.2);

INSERT INTO sub_firewarden VALUES ('I0001', 122344, 33.1);
INSERT INTO sub_firewarden VALUES ('I0002', 202340, 55.3);
INSERT INTO sub_firewarden VALUES ('I0003', 123455, 77.2);
INSERT INTO sub_firewarden VALUES ('I0004', 122344, 33.1);
INSERT INTO sub_firewarden VALUES ('I0005', 202340, 55.3);
INSERT INTO sub_firewarden VALUES ('I0006', 123455, 77.2);
```

Note how the same member of staff can occupy both roles

Question: Why don't I store the bonus field in the super-class table?

```
Because the value of the bonus is different for each of the roles.  A
firewarden gets a different amount to the firstaider.
```

Question: I want to list all members of staff – common fields only, but including the bonus field.
Answer

```
SELECT super_staff.staff_id, staff_name, age, salary, bonus
FROM super_staff, sub_firstaider
WHERE super_staff.staff_id = sub_firstaider.staff_id
UNION
SELECT super_staff.staff_id, staff_name, age, salary, bonus
FROM super_staff, sub_firewarden
WHERE super_staff.staff_id = sub_firewarden.staff_id;
```

What is the problem with the records that are output?

```
It appears as if some staff members appear twice and it is not clear which
entry is for which role.
```

How can we solve this problem?

```
Try adding a field called Role to each subclass table, which, in the
firstaider subclass table will contain FA and in the firewarden subclass
table will contain FW.

Then rewrite the above query so that it also displays the Role field.
```