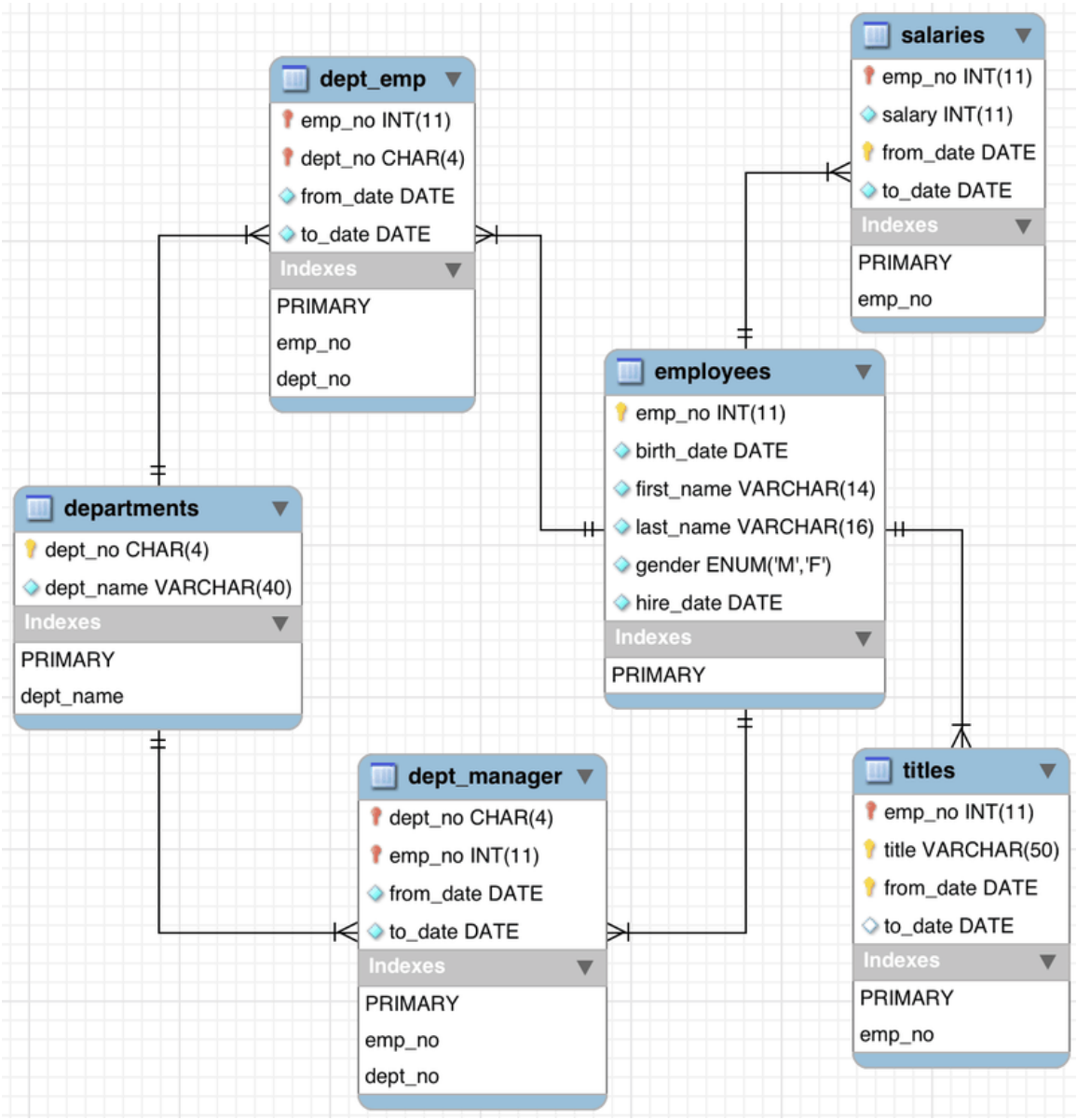
**TEST 1 SAMPLE SOLUTION**
CIS 451/551 Database Processing

1. Consider the EMPLOYEES database described by the following Crowsfoot diagram.



Write SQL queries for this schema which

a) For each department, list the department name and the number of employees working for that department that have the title "Barge Scraper". Those with zero such employees need not be listed.

```
SELECT dept_name, COUNT(*)
FROM departments JOIN dept_emp USING(dept_no)
     JOIN employees USING (emp_no)
     JOIN titles USING(emp_no)
WHERE title LIKE 'Barge Scraper'
     GROUP BY 1
```

b) List all employees who work in the department with the most employees.

```
SELECT first_name, last_name
FROM employees JOIN dept_emp USING(emp_no)
     JOIN
          (SELECT dept_no, COUNT(emp_no) AS num_emp
          FROM dept_emp
          GROUP BY dept_no
          ORDER BY num_emp
          LIMIT 1) AS c
     USING(dept_no)
```

c) List all employees who are not managers who have a larger salary than some manager (and show the manager's name).

```
SELECT CONCAT(ee.first_name, ' ', ee.last_name) AS emp_name,
se.salary AS emp_salary,
     CONCAT(em.first_name, ' ', em.last_name) AS man_name,
sm.salary AS man_salary
FROM employees ee CROSS JOIN employees em
     JOIN salaries se ON ee.emp_no=se.emp_no
     JOIN salaries sm ON em.emp_no=sm.emp_no
WHERE ee.emp_no NOT IN (SELECT emp_no FROM dept_manager)
     AND
     em.emp_no IN (SELECT emp_no FROM dept_manager)
     AND
     se.salary>sm.salary
```

d) Show all managers who do not work in the department that they manage.

```
SELECT e.first_name, e.last_name, dm.dept_no
FROM employees e JOIN dept_manager dm USING(emp_no)
WHERE NOT EXISTS
```

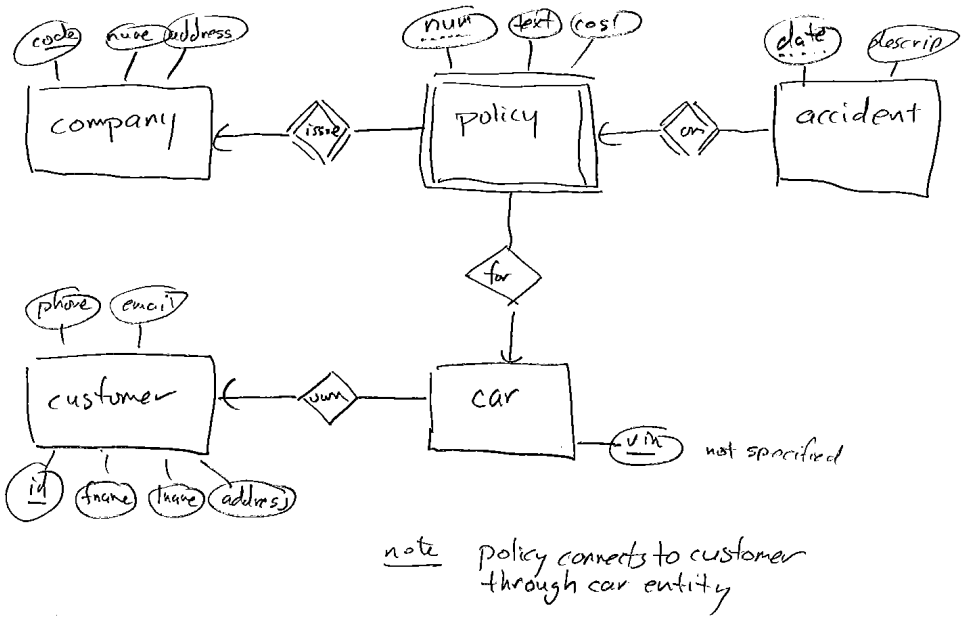
```
     (SELECT * FROM dept_emp de WHERE de.emp_no=e.emp_no
AND de.emp_no=dm.dept_no)
```

e) (**551**) List the average salary for the employees of each department. Show the department name and include a 0 for departments with no employees.

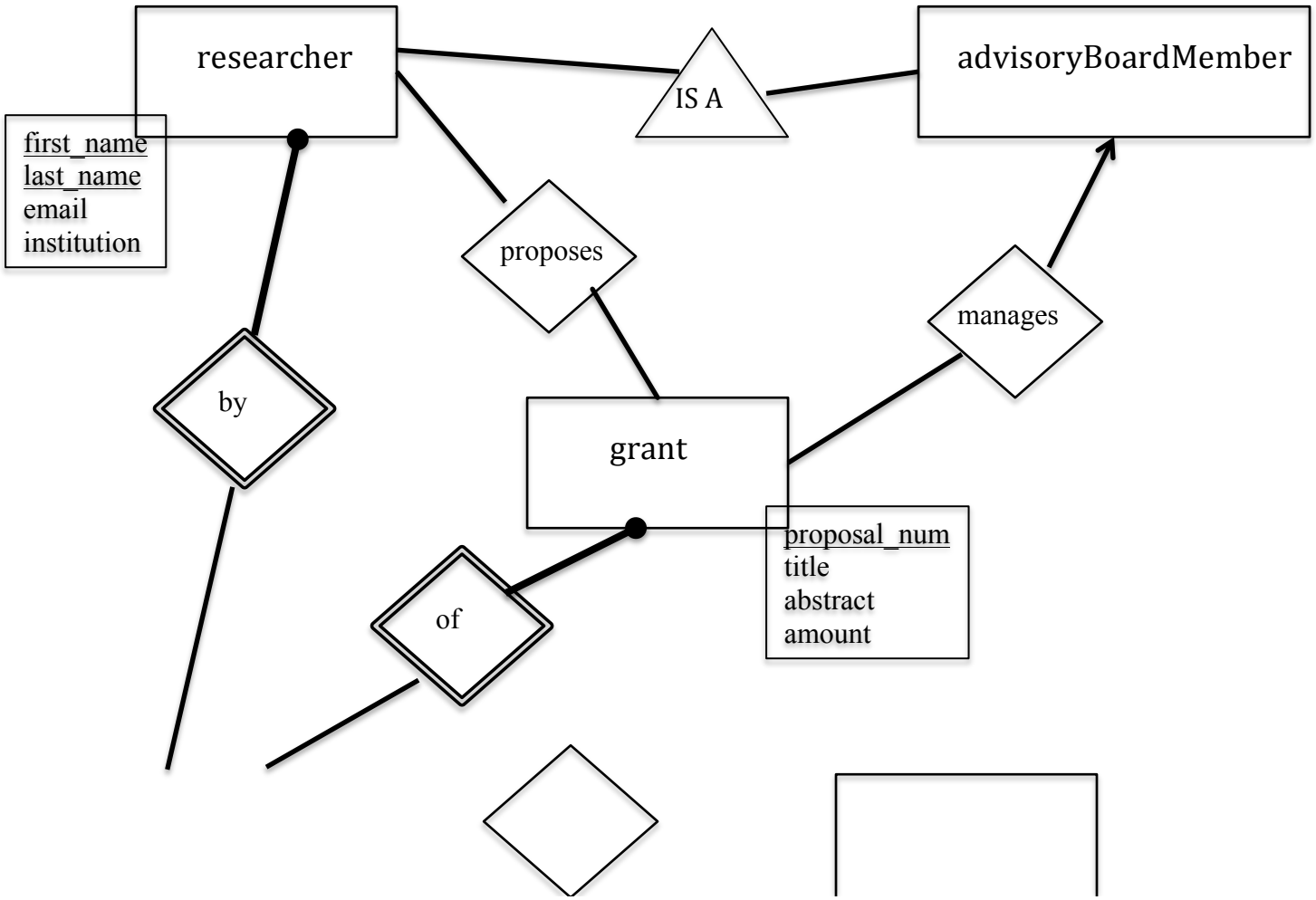
```
SELECT dept_name, AVG(salary)
FROM salaries JOIN employees USING(emp_no)
     JOIN dept_emp USING (emp_no)
     RIGHT JOIN departments USING(dept_no)
     GROUP BY dept_name
```

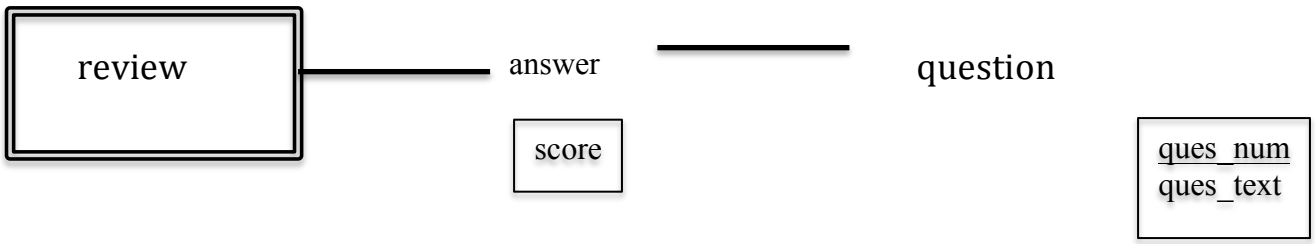
2. Draw an ER Diagram for the CAR_INSURANCE problem.
   - There are several insurance **companies**, with a unique code, they also have a name and an address.
   - There are many **customers**, with customer id, first and last names, address, phone, and email.
   - Customers can own many several **cars**, but a car is owned by exactly one customer.
   - Customers purchase **policies** from the companies for their cars. A policy has a text, a policy number, and a cost. A policy involves one car, one customer, and one company.
   - Two different companies may issue a policy with the same policy number (so **policy** is weak).
   - **Accidents** need to be recorded. Accidents have a date (partial key), a description, and are charged to (and owned by) a policy.

   Be sure to use the Chen ER notation from the text or the Crowsfoot style as used by MySQLWorkbench.




3. Convert the GRANT ER Diagram to a relational schema. Indicate primary keys and foreign keys.






NOTES

- review is a weak entity, owned by "by" and "of"
- attributes are in the small boxes attached to the entities
- the ovals on the lines next to researcher and grant indicate mandatory membership, as where the text uses curved line arrow tips.
- each advisory board member is a researcher

RESEARCHER: first_name, last_name, email, institution

ADVISORYBOARDMEMBER: first_name, last_name
*first_name, last_name foreign key to RESEARCHER*

GRANT: proposal_num, title, abstract, amount, first_name, last_name
*first_name, last_name foreign key to ADVISORYBOARDMEMBER*

QUESTION: ques_num, ques_text

PROPOSES: proposal_num, first_name, last_name
*first_name, last_name foreign key to RESEARCHER*
*proposal_num foreign key to GRANT*

REVIEW: first_name, last_name, proposal_num
*first_name, last_name foreign key to RESEARCHER*
*proposal_num foreign key to GRANT*

ANSWER: first_name, last_name, proposal_num, ques_num, score
*first_name, last_name, proposal_num foreign key to REVIEW*
*ques_num foreign key to QUESTION*