Ryan Ballenger
CSCIE66 HW#1 Part 1

1. a) Describe how you would modify Figure 1-1 to reflect the fact that every course is taught by *at least one* professor. (Describing the change in words is fine, although you may use a diagram if you prefer.)

I would thicken the line between “Course” and “Teaches”. This participation constraint means the course must be taught by at least one professor.

1. b) Describe how you would modify the original version of Figure 1-1 to reflect the fact that every course must be taught by *at most one* professor. (Describing the change in words is fine, although you may use a diagram if you prefer.)

I would add an arrow pointing from “Teaches” to “Professor”. This creates a many to one relationship where each course can be taught by at most one professor.

1. c) Describe how you would modify the original version of Figure 1-1 to reflect the fact that every professor must teach *exactly one* course. (Describing the change in words is fine, although you may use a diagram if you prefer.)

I would add an arrow from “Teaches” to “Course” and bolden the line between “Professor” and “Teaches”. The arrow means each professor can teach at most one course and the boldened line means each professor must teach at least one course, which together means every professor must teach exactly one course.

1. d) Consider your answer to part b -- the ER diagram for the situation in which every course is taught by *at most one* professor. If we converted that ER model to a relational schema, would the following be an acceptable schema for a relation used to capture the Teaches relationship set?*Teaches(<u>professor</u>, <u>course</u>, semester)* where *professor* is a foreign key referring to *Professor(id)*, *course* is a foreign key referring to *Course(name)*, and the primary-key attributes of *Teaches*are underlined. Explain your answer briefly.

No, because the same course could be offered both fall and spring semester, which is very common. Therefore, professor and course are not primary key attributes since they don’t specify which semester in this case.

2.a) *(2 points)* At least one of the relationship sets captures a many-to-one relationship. Which one(s)? In your answer, you should specify the direction of the relationship(s) (e.g., "________ is a many-to-one relationship from _________ to ___________").
Borrows is a many-to-one relationship from Customer to Account.
HasType is a many-to-one relationship from Account to AccountType.

2.b) ***(2 points)*** **Describe all constraints on relationships that are specified by the diagram. Use words that describe the problem domain (e.g., "Each course meets in at most one room...") rather than technical terminology.**
Each Customer has at most one Loan.
Each Loan must have at least one Customer.
Each Account must be owned by at least one Customer.
Each Account must have exactly one AccountType.

2.c)***(4 points)*** **Transform this diagram into a relational schema. Give the schema of each relation in the form *relation_name(attr_name1, attr_name2,...)*. When appropriate, you should combine relations as discussed in lecture; if you do combine two relations, you should briefly explain why it makes sense to do so. Indicate the primary-key attribute(s) of each relation by putting an underscore character (_) on either side of the attribute name(s). For example, if you had a key called id, you would write it as _id_. In addition, you should specify each relation's foreign-key attribute(s) (if any) and state the associated referential-integrity constraints. An example of specifying a referential-integrity constraint for a university database is: "each value of the advisor attribute must match a value of the id attribute from the Faculty relation".**

Loan(_id_, principal, interest, interest_rate)

Customer(_id_, customer_name, customer_address, loan_id)

> I combined customer and borrows because I assume a significant amount of customers have a loan and there will not be a wasteful amount of NULLs in loan_id. This saves space by eliminating a Borrows table and works because each customer has at most one loan.
>
> **Note:** If few customers have a loan, a Borrows(_customer_id_, loan_id,) table should be made and loan_id should be dropped from Customer.
>
> loan_id is a foreign key to Loan._id_. Each value of the loan_id must match a value of _id_ in the Loan relation.

Account(_account_id_, balance, name, interest_rate)

> Account and account type are combined since each account must have exactly one account type.

Owns(_account_id_, _customer_id_)

> Owns cannot be combined with Account because multiple customers can own an account.
>
> _account_id_ is a foreign key to Account._account_id_. Each value of the _account_id_ must match a value of _account_id_ in the Account relation.

_customer_id_ is a foreign key to Customer._id_. Each value of the _customer_id_ must match a value of _id_ in the Customer relation.

3.a.) What is the Cartesian product of R and S?

R.a	**R.b**	**S.c**	**S.b**	**S.a**
1	2	3	2	1
1	2	5	4	3
1	2	9	8	7
3	4	3	2	1
3	4	5	4	3
3	4	9	8	7
5	6	3	2	1
5	6	5	4	3
5	6	9	8	7

3.b) What is the natural join of R and S?

a	**b**	**c**
1	2	3
3	4	5

3.c) What is the left outer join of R and S?

a	**b**	**c**
1	2	3
3	4	5
5	6	NULL

3.d) What is the right outer join of R and S?

a	**b**	**c**
1	2	3
3	4	5
7	8	9

3.e) What is the full outer join of R and S?

a	b	c
1	2	3
3	4	5
5	6	NULL
7	8	9

4.

Note: For a,b,c I used indenting and spacing to make the algebra more readable but assuming the compiler ignores spacing, it should not affect the function of each.

4.a) problem 5 (Oscars won by Cate Blanchett)

```
$\pi_{\text{Movie.name, type, Oscar.year}}($
    $\sigma_{\{\text{Person.name="Cate Blanchett"}\}}$ (
        Person THETA-JOIN$_{\{\text{Person.id=person\_id}\}}$(
            Movie THETA-JOIN$_{\{\text{Movie.id=Oscar.movie\_id}\}}$ Oscar)
        )
    )
)
```

4.b) problem 8 (films that won both Best Picture and Best Director)

Note: The first Oscar’s attributes are named One.attribute after the rename.

```
$\pi_{\text{Oscar.year, One.name}}($
    $\sigma_{\{\text{One.type = "BEST-PICTURE", Oscar.type="BEST-DIRECTOR"}\}}$ (
        Oscar THETA-JOIN$_{\{\text{Oscar.movie\_id=One.id}\}}$(
            $\rho_{\text{One}}$(Movie THETA-JOIN $_{\{\text{id=movie\_id}\}}$ Oscar)
        )
    )
)
```

4.c) problem 13 (Oscars won by the top 25 grossing films; you do *not* need to sort the results as you do in the SQL query)

$$\pi_{earnings\_rank,\ name,\ type}\Big($$

$$\qquad\sigma_{\{earnings\_ranking\ <=\ 25\}}\Big($$

$$\qquad\qquad\text{Movie LEFT-OUTER-JOIN}_{\{id=movie\_id\}}\ \text{Oscar}$$

$$\qquad\Big)$$

$$\Big)$$