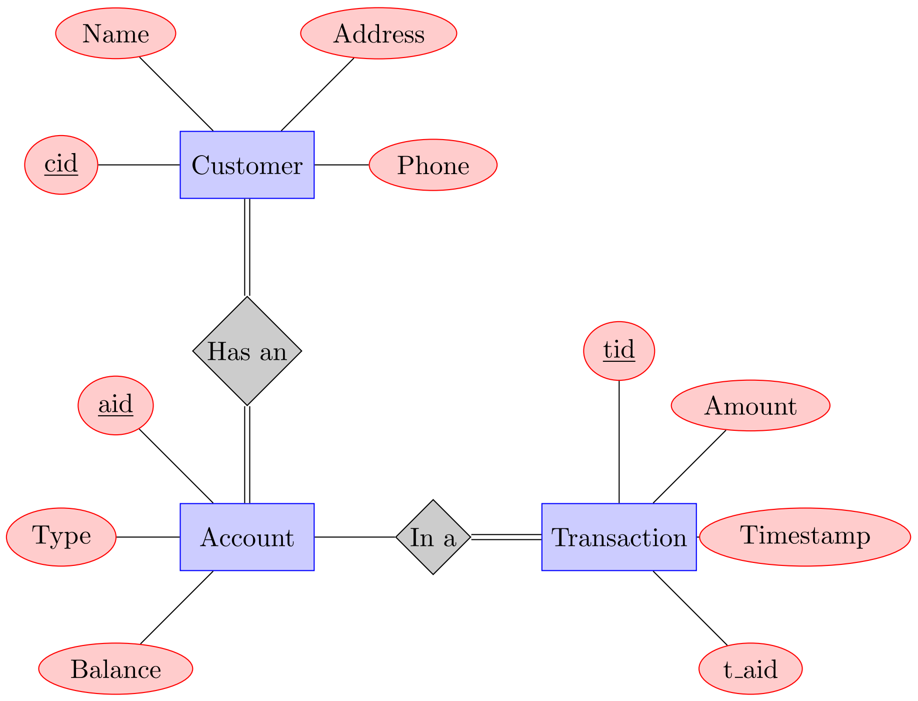
Introduction to Database Systems

Problem Set 1 Solution

1. 

2. (a) $\pi_{c\_name}(customer \bowtie_{cid=a\_cid} \sigma_{bal>\$N}(account))$
   (b) $\pi_{aid}(\sigma_{count>n}({}_{aid}G_{count(*)}(\sigma_{type='Savings'}(account))))$
   (c) $\pi_{c\_name}(\sigma_{amount>\$N}(transaction) \bowtie_{t\_aid=aid} account \bowtie_{a\_cid=cid} customers)$

3. The first plan is most efficient because it filters down account before joining it with customer. (b) None of the operators are commutative, hence it is the fastest. The count needs to be completed before we can select from its output. For (c), this plan is the fastest only if joining the filtered transaction (t') by account compares and produces fewer tuples that account and customer. If joining t' with accounts produces t_a output tuples, and joining account by customer has an output size of t_b, then the plan above has a cost of $t_a \times sizeof(customer)$. Joining $customer \bowtie account \bowtie \sigma(transaction)$ will have a cost of $t_b \times sizeof(t')$. In practice, we would estimate sizeof(t'), t_a, and t_b, and pick the ordering with the lowest cost.

4. (a)
```
SELECT c_name
FROM accounts, customers
WHERE a_balance > N and a_cid = cid;
```
   (b)
```
WITH customer_cnts as (
SELECT aid, count(*) as cnt
FROM account
WHERE type='Savings'
                        GROUP BY aid)
SELECT aid
FROM customer_cnts
WHERE cnt > n;
```
   (c)
```
SELECT c_name
FROM accounts, transactions, customers
```

```
    WHERE cid = a_cid and t_aid = aid and transaction.amount > N;
```

5. The functional dependencies are:
   (a) cid $\Longrightarrow$ c_name, c_address, c_phone
   (b) aid $\Longrightarrow$ a_type, a_balance
   (c) tid $\Longrightarrow$ t_timestamp, t_amt, t_aid
   (d) (cid, aid, tid) $\Longrightarrow$ (c_name, c_addr, c_phone, a_type, a_balance, t_aid, t_timestamp, t_amt) (optional)

6. Primary keys are underlined.
   Start schema: (cid, c_name, c_addr, c_phone, aid, a_type, a_balance, tid, t_aid, t_timestamp, t_amt)
   Start functional dependency: (cid, aid, tid) $\Longrightarrow$ (c_name, c_addr, c_phone, a_type, a_balance, t_aid, t_timestamp, t_amt)

   (a) cid $\Longrightarrow$ c_name, c_address, c_phone
   (cid, c_name, c_addr, c_phone)
   (cid, aid, a_type, a_balance, tid, t_aid, t_timestamp, t_amt)

   (b) aid $\Longrightarrow$ a_type, a_balance (cid, c_name, c_addr, c_phone)
   (aid, a_type, a_balance)
   (cid, aid, tid, t_aid, t_timestamp, t_amt)

   (c) tid $\Longrightarrow$ t_timestamp, t_amt, t_aid (cid, c_name, c_addr, c_phone)
   (aid, a_type, a_balance)
   (tid, t_aid, t_timestamp, t_amt)
   (cid, aid, tid)

   (d) Bonus: tid is redundant in the last relation –it is a foreign key (1:N) relationship. Hence, our final schema is:
   (cid, c_name, c_addr, c_phone)
   (aid, a_type, a_balance)
   (tid, t_aid, t_timestamp, t_amt)
   (cid, aid)

   The schema in (d) is in BCNF because it has no redundancy. (c)'s layout does not have this property.