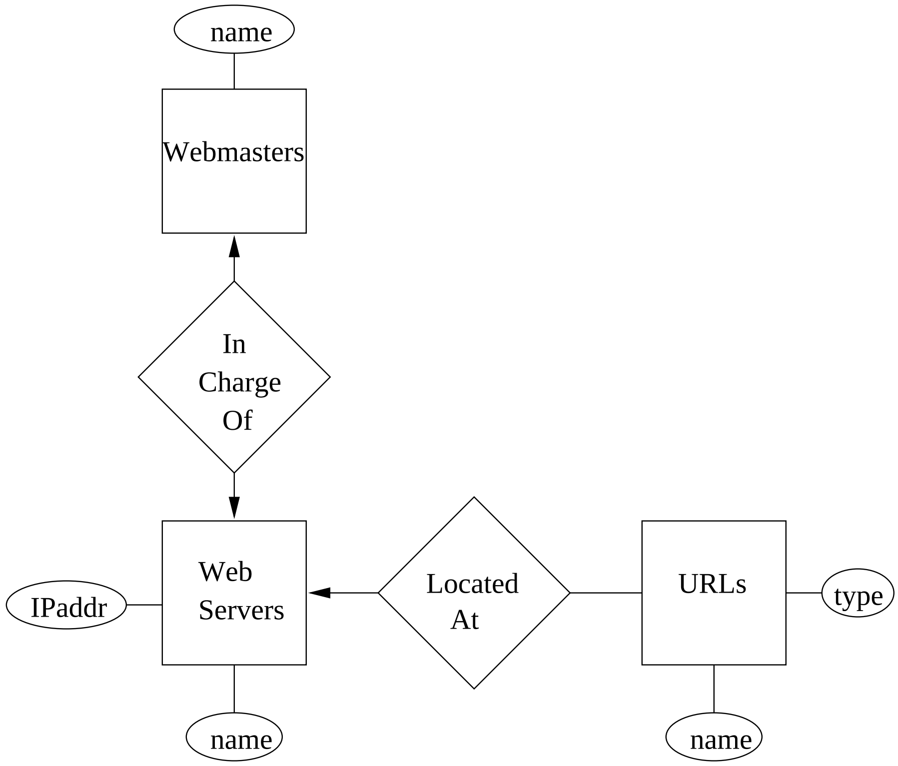
PART B (BASIC)

1. (12 pts.) Below is an E/R diagram that is a first attempt at a design for a database that will include information about Web servers and the URL's of pages at those servers. For each web server, we want to represent its IP address (e.g., 171.64.75.107), its registered name (e.g., www-db.stanford.edu), and the name of its webmaster. We assume a webmaster can be in charge of only one server. A URL has a name, such as `www-db.stanford.edu/pub/foo.html`, and a type, HTML in this example.



   (a) Give the two most important things that are wrong with this design.

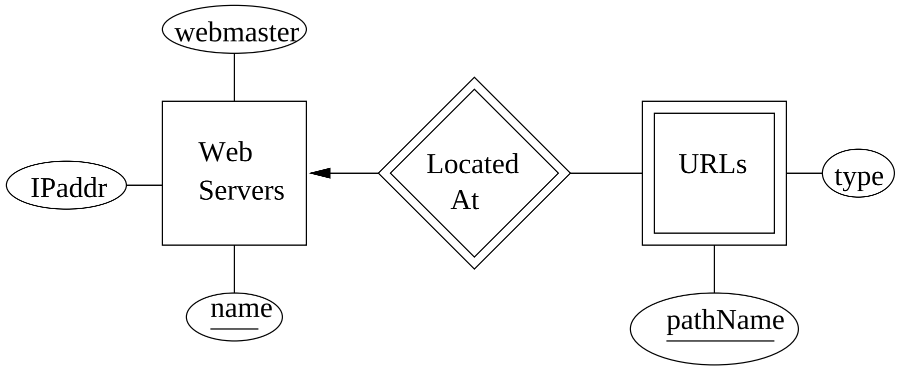
   (b) Explain how you would fix the problems, and give an improved E/R diagram.

Answer:

1) The Webmaster concept, being simply a name, is needless. If all we ever care about is the Webmaster's name, then it would be better to make that name an attribute of the server entity set.

2) URL's and server names are redundant, because the former includes the latter. As the design stands, our database could store a file whose URL says it is at one server, while the *At* relationship says it is at another server. We need to replace URL's by path names, and let the complete URL be inferred by the weak-entity-set mechanism and the associated host name.

Note that there are some unrealistic elements in this problem, e.g., an IP address or host name can really refer to multiple servers, and conversely, a server could have multiple names or IP addresses. Likewise, it is unlikely that a webmaster is uniquely identified by their name. However, the following design is consistent with the assumptions given in the problem statement:



2. (8 pts.) Suppose we have a relation ABC with functional dependencies $A \rightarrow B$ and $B \rightarrow C$.
   (a) Show that this relation is not in Boyce-Codd Normal Form (BCNF).
   (b) Add one more FD so that the relation now is in BCNF.
   (c) Is it possible to add one FD to $A \rightarrow B$ and $B \rightarrow C$ to make the relation be in third normal form but not in BCNF? Demonstrate or explain why not.

Answer:

(a) The only key is A, so the FD $B \rightarrow C$ has a left side that is not a superkey.

(b) If we add $B \rightarrow A$, then B becomes a key, along with A, and all FD's have either A or B on the left and so are no longer BCNF violations.

(c) It is not possible. One might try to reason it out, e.g., by observing that in order for $B \rightarrow C$ to be a BCNF violation but not a 3NF violation, we would have to make C prime and B not a key. That reasoning is tricky and doesn't address the possibility that $B \rightarrow C$ might *not* violate BCNF, but the third FD does. Fortunately, there are only four nontrivial FD's that do not already follow from the given ones. We can look at all four and check that none leave us in the state of 3NF-but-not-BCNF:
   i. $B \rightarrow A$ puts us in BCNF, as above.
   ii. $C \rightarrow A$ makes all three attributes keys, and so also yields BCNF.
   iii. $C \rightarrow B$ leaves only A a key, so the relation is not in 3NF.
   iv. $BC \rightarrow A$ makes A and B the keys, and thus the relation is in BCNF.

PART B (ADVANCED)

3. (6 pts.) Suppose we have a set of integers $S(A)$, i.e., a unary relation whose attribute A is of type `INT`, and which is guaranteed to have no duplicates. Show that for any fixed k we can write an expression of relational algebra that produces a relation consisting of only the kth largest integer in S.

Answer: Perhaps the easiest way is to do an induction on k. We construct S_i to be S minus the i largest values, and T_i to be the set containing only the ith largest value.

Basis: $S_0 = S$.

Induction: $T_i = S_{i-1} - \pi_{A1}(S_{i-1}(A1) \underset{A1<A2}{\bowtie} S_{i-1}(A2))$ and $S_i = S_{i-1} - T_i$.

4. (4 pts.) Despite some similarity of syntax, the group-by operation in OQL has a significant power that group-by in SQL does not have. What is that power?

   Answer: In SQL, grouping is only by values of attributes. In OQL, it is possible to group by any function of the objects involved.

5. (15 pts.) Suppose we have global predicates $r(X, Y)$ and $b(X, Y)$, which you may think of as red and blur arcs in a graph, if you like. There are three views on these predicates defined as follows:

```
v1(X,Y) :- r(X,Y) & b(Y,Z)
v2(X,Y) :- b(Z,X) & r(X,Y)
v3(X,Y) :- b(X,Y) & r(Y,Z)
```

   Intuitively, v_1 is a red arc that can be followed by a blue arc; v_2 is a red arc that can be preceded by a blue arc, and v_3 is a blue arc that can be followed by a red arc. We wish to find a *maximally contained rewriting* of the query:

```
Q: answer(A,D) :- r(A,B) & b(B,C) & r(C,D)
```

   that is, a set of conjunctive queries $Q_1, Q_2, \ldots, Q_n$, whose subgoals may use only the views v_1, v_2, and v_3, and such that:

   i. Each Q_i is contained in Q,

   ii. No Q_i is contained in another Q_j, and

   iii. It is not possible to add an additional conjunctive query to the set and still satisfy (*i*) and (*ii*).

   Your questions:

   (a) Find the maximally contained rewriting of the query Q using the views v_1, v_2, and/or v_3. Justify your choices for partial credit.

   (b) Is the maximally contained rewriting of Q equivalent to Q. Explain why or why not.

   Answer:

   (a) While there are many ways to approach this problem, perhaps the simplest is to observe that no view can cover more than one subgoal of the query. Thus, we can only use v_1 and/or v_2 to cover r and v_3 to cover b. All combinations actually work when the proper variables are used, so we get the following four solutions:

```
answer(A,D) :- v1(A,B) & v3(B,C) & v1(C,D)
answer(A,D) :- v1(A,B) & v3(B,C) & v2(C,D)
answer(A,D) :- v2(A,B) & v3(B,C) & v1(C,D)
answer(A,D) :- v2(A,B) & v3(B,C) & v2(C,D)
```

(b) Unfortunately, the problem did not state whether a closed- or open-world assumption should be taken, so we wound up accepting either. If you took an open-world assumption, then it is possible that none of the views contain any tuples, even if the global predicates have lots of tuples that lead to lots of answers for Q. Thus, the answer is "no." On the other hand, if you made the closed world assumption, where the definitions of the views reflect tuples actually present, then the second solution above is equivalent to Q. In that case, the answer is "yes."

6. (15 pts.) A *generalized dependency* on a relation R can be written in the form

$$Head \texttt{ :- } R(X_{11}, \ldots, X_{1k}) \ \& \ \cdots \ \& \ R(X_{n1}, \ldots, X_{nk})$$

where the body consists of $n \geq 1$ subgoals, each with predicate R and arbitrary variables as arguments (i.e., variables may repeat in any number of positions, in one or several subgoals). The head is either:

i. Another atom consisting of the predicate R and some variables. We call this dependency *tuple-generating*. If all the variables of the head are present in the body, then the dependency is *full*; if the head has variables not appearing in the body, then the dependency is *embedded*.

ii. An equality between two variables that appear in the body. We call such a dependency *equality-generating*.

For instance, in the relation $R(A, B, C, D)$, the generalized dependency

```
X=Y :- R(U,V,X,W) & R(U,V,Y,Z)
```

is an equality-generating dependency that is equivalent to the functional dependency $AB \rightarrow C$.

A generalized dependency asserts that whenever we assign values to the variables of the body so that all subgoals become tuples of R, then

- If the head is an equality, then it must be that the two equated variables were assigned the same value in order for all the subgoals to be in R.
- If the head is $R(Y_1, \ldots, Y_k)$, then we can can find values for any of the Y's that do not appear in the body, such that the tuple formed from the values of $Y_1, \ldots, Y_k$ is in R.

Your questions:

(a) Write a generalized dependency that is equivalent to the multivalued dependency $AB \twoheadrightarrow C$ in the relation $R(A, B, C, D)$.

(b) Write a generalized dependency that is equivalent to the join dependency

$$AB \bowtie BC \bowtie CD$$

in the relation $R(A, B, C, D, E)$; i.e., the projection of the relation R onto AB, BC, and CD, followed by the join of those relations yields the projection of original relation onto $ABCD$.

(c) Give an example of an equality-generating dependency that is *not* equivalent to any functional dependency. Explain what constraint your dependency expresses, and why it is not a FD.

Answer:

(a)
```
R(A,B,C,D) :- R(A,B,C,D1) & R(A,B,C1,D)
```

(b)
```
R(A,B,C,D,E) :- R(A,B,C1,D1,E1) & R(A2,B,C,D2,E2) & R(A3,B3,C,D,E3)
```

(c) There were many possible answers, of course. Perhaps the simplest is

```
A=B :- R(A,B)
```

This dependency says that all tuples must be of the form (x, x). However, no FD can enforce this condition. For instance, $R = \{(0, 1)\}$ is a relation that satisfies every FD on R (as any one-tuple relation must), yet does not satisfy the equality-generating dependency above.