Assignment 3
CIS 451/551 Fall 2005
due 5 pm, November 1st

Chapter 6 Problems

(10) 1. Explain the differences between primary keys, candidate keys, and superkeys. If each is a set of keys within a table, how are they related to each other?

Answer  A superkey is a set of one or more attributes that, taken collectively, allows us to identify uniquely an entity in the entity set. A superkey may contain extraneous attributes. If K is a superkey, then so is any superset of K. A superkey for which no proper subset is also a superkey is called a candidate key. It is possible that several distinct sets of attributes could serve as candidate keys. The primary key is one of the candidate keys that is chosen by the database designer as the principal means of identifying entities within an entity set.

(15) 2. Construct an E-R diagram and an associated table (with keys) for a hospital with a set of doctors that has any number of patients associated with them. Also include a log of various tests and examinations for each patient conducted/ordered by the patient’s doctor.

Answer  E-R Diagram Figure 1.

Table:
patients (patient-id, name, insurance, date-admitted, date-checked-out)
doctors (doctor-id, name, specialization)
test (testid, testname, date, time, result)
doctor-patient (patient-id, doctor-id)
test-log (testid, patient-id)
performed-by (testid, doctor-id)

Figure 1: Hospital E-R diagram.
3. What is the difference between a weak and strong entity set? What is the purpose of having a weak entity set?

Answer  A strong entity set has a primary key. All tuples in the set are distinguishable by that key. A weak entity set has no primary key unless attributes of the strong entity set on which it depends are included. Tuples in a weak entity set are partitioned according to their relationship with tuples in a strong entity set. Tuples within each partition are distinguishable by a discriminator, which is a set of attributes.

There are several reasons for weak entity sets:

- We want to avoid the data duplication and consequent possible inconsistencies caused by duplicating the key of the strong entity.
- Weak entities reflect the logical structure of an entity being dependent on another entity.
- Weak entities can be deleted automatically when their strong entity is deleted.
- Weak entities can be stored physically with their strong entities.

Chapter 7 Problems

4. Use Armstrong’s axioms to prove the soundness of the decomposition of the rule: if $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds.

Answer  Given: $\alpha \rightarrow \beta\gamma$

$\beta\gamma \rightarrow \beta$ reflexivity rule
$\alpha \rightarrow \beta$ transitivity rule
$\beta\gamma \rightarrow \gamma$ reflexivity rule
$\alpha \rightarrow \gamma$ transitivity rule

5. Use the definition of functional dependency to argue that each or Armstrong’s axioms (reflexivity, augmentation, and transitivity) is sound. The definition of functional dependency is: $\alpha \rightarrow \beta$ holds on $R$ if in any legal relation $r(R)$, for all pairs of tuples $t_1$ and $t_2$ in $r$ such that $t_1[\alpha] = t_2[\alpha]$, it is also the case that $t_1[\beta] = t_2[\beta]$.

Answer  Reflexivity rule: if $\alpha$ is a set of attributes, and $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$. Assume $\exists t_1, t_2$ such that $t_1[\gamma\alpha] = t_2[\gamma\alpha]$.

$t_1[\beta] = t_2[\beta]$ since $\beta \subseteq \alpha$
$\alpha \rightarrow \beta$ definition of FD
Augmentation rule: if $\alpha \rightarrow \beta$, and $\gamma$ is a set of attributes, then $\gamma\alpha \rightarrow \gamma\beta$.
Assume $\exists t_1, t_2$ such that $t_1[\gamma\alpha] = t_2[\gamma\alpha]$.

$$
t_1[\gamma] = t_2[\gamma] \quad \gamma \subseteq \gamma\alpha
$$

$$
t_1[\alpha] = t_2[\alpha] \quad \alpha \subseteq \gamma\alpha
$$

$$
t_1[\beta] = t_2[\beta] \quad \text{definition of } \alpha \rightarrow \beta
$$

$$
t_1[\gamma\beta] = t_2[\gamma\beta] \quad \gamma\beta = \gamma \cup \beta
$$

$$
\gamma\alpha \rightarrow \gamma\beta \quad \text{definition of FD}
$$

Transitivity rule: if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$.
Assume $\exists t_1, t_2$ such that

$$
t_1[\alpha] = t_2[\alpha]
$$

$$
t_1[\beta] = t_2[\beta] \quad \text{definition of } \alpha \rightarrow \beta
$$

$$
t_1[\gamma] = t_2[\gamma] \quad \text{definition of } \alpha \rightarrow \gamma
$$

$$
\alpha \rightarrow \gamma \quad \text{definition of FD}
$$

(15) 6. Please get a lossless-join decomposition into BCNF for the Publication schema:

Publication(Title, Author, Book_type, Author_affil, ISBN)

The functional dependencies holds for this schema:

$$
\text{Title} \rightarrow \text{Author}, \text{Book_type}
$$

$$
\text{Book_type,Author_affil} \rightarrow \text{ISBN}
$$

$$
\text{Author} \rightarrow \text{Author_affil}
$$

$$
\text{ISBN} \rightarrow \text{Title}
$$

**Answer** First, compute the closure over the dependency rules and find the candidate keys (relations that relate to all relations), which are: Title, ISBN, Author, Book_type, and book_type Author_affil.

$\text{Author} \rightarrow \text{Author_affil}$ is nontrivial and the left hand side is not a superkey. By the algorithm of BCNF decomposition (Figure 7.12, 5th Edition), we derive the relations $(\text{Title}, \text{Author}, \text{Book_type}, \text{ISBN}), (\text{Author, Author_affil})$. This is in BCNF.

(15) 7. Please get a lossless-join and dependency-preserving decomposition into 3NF for the Publication schema and functional dependencies in the last problem.

**Answer** There are two acceptable answers for this solution. The first solution is that by analyzing $R$ in 3NF, we can verify that $R$ is already in 3NF. Within the functional dependencies: Title, Book_type Author_affil, and ISBN are all candidate keys. Though Author in the functional dependence Author $\rightarrow$ Author_affil is not a key, Author_affil is contained in the candidate key Book_type Author_affil.
The second solution is to use the 3NF decomposition algorithm (figure 7.13). Initially, we determine that the given functional dependencies form a canonical cover $F_c$. Next, we apply the 3NF decomposition algorithm. The result is:

Publication’ = \{(Title,Author,Book_type),\( (Book\_type,Author\_affil,ISBN),\)
\( (Author,Author\_affil),(ISBN,Title)\}\)

Since schema \((Title,Author,Book\_type)\) contains a candidate key. Therefore \(Publication'\) is a third normal form dependency-preserving lossless-join decomposition.