Contents

Chapter 1

Functional Dependency

1.1 Relation

Definition 1.1. A *relation* is an ordered pair (S,R), where S is an n-tuple of names of attributes, and R is a set of n-tuples with values for the attributes as described by S.

Given $T \in R$ and $S = (s_1, s_2, ...s_n)$, we denote the value for attribute s_1 in T as $T(s_1)$.

1.2 Functional Dependency

Definition 1.2. Given R and $S = (X, Y)$, we say that X determines Y, denoted $X \rightarrow Y$, if $T_1(X) = T_2(X)$ implies $T_1(Y) = T_2(Y)$ for any $T_1, T_2 \in R$, and we call this a *functional dependency.*

1.3 Armstrong's Axioms

1. *Reflexivity*: if $Y \subseteq X$ then $X \rightarrow Y$
2. *Augmentation*: if $X \rightarrow Y$ then $XZ \rightarrow YZ$ for any Z
3. *Transitivity*: if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

1.4 Closure

Definition 1.3. The *closure* of a set F of functional dependencies, denoted $F+$, is the set of all functional dependencies that can be inferred from those in F.

1.5 Minimal Cover

Definition 1.4. The *minimal cover* G of a set F of functional dependencies, is the smallest set such that $G+ = F+$.

Chapter 2

Normal Forms

2.1 First Normal Form (1NF)

Definition 2.1. A *superkey* of a relation S, R is a set of attributes X such that $t_1(X) = t_2(X)$ if and only if $t_1 = t_2$. Such attributes are said to be *prime*.

A superkey is said to be *minimal* if it has the least number of attributes required to meet this condition. Such a minimal superkey is called a candidate key.

Definition 2.2. A set of relations is in *First Normal Form* if every relation has a minimal superkey.

2.2 Second Normal Form (2NF)

Definition 2.3. A *partial dependency* is a dependency of a non-prime attribute on a proper subset of a candidate key.

Definition 2.4. A set of relations is in *Second Normal Form* if it is in 1NF and it contains no partial dependencies.

2.3 Third Normal Form (3NF)

Definition 2.5. A *trivial dependency* is a dependency $X \to Y$ where $Y \subseteq X$.

Definition 2.6. A *transitive dependency* is a dependency inferred from the transitive axiom. If $X \to Y$ is a transitive dependency, we say Y is *transitively dependendent* on X, otherwise Y is *directly dependent* on X.

Definition 2.7. A set of relations is in *Third Normal Form* if it is in 2NF and all functional dependencies $X \to Y$ are trivial, or X is a superkey, or all attributes $a \in (X - Y)$ are prime.

2.4 Boyce-Codd Normal Form (BCNF)

Definition 2.8. A set of relations is in *Boyce-Codd Normal Form* if it is in 2NF and all functional dependencies $X \to Y$ are trivial or X is a superkey.

Chapter 3

Relational Algebra

Given a relation (S, R), we define a formalism for retrieving information.

3.1 Select

The *selection* operator denoted σ takes two paramters: a set of attributes in S and a relation (S, R), and returns a new relation (S', R') in which S' contains only the specified attributes of S and R' contains modified tuples with only values corresponding to the attributes of S'. We denote this as σ_{X}Y where X is the set of attributes to keep and Y is the relation.

3.2 Project

The *projection* operator denoted π takes two parameters: a set of conditions on the attributes of S and a relation (S, R), and returns a new relation (S, R') where $R' \subseteq R$ is the set of all tuples that meet the specified conditions. We donte this as π_{X}Y where X is the set of conditions and Y is the relation.

3.3 Renaming Attributes

It is often useful to rename attributes and save intermediate relations. We do this with the $\leftarrow$ operator as in the following example.

Stuff(Noms,NotNoms) $\leftarrow \sigma_{\mathrm{Edibles,Inedibles}}$ Items

This example selects the Edibles and Inedibles attributes of the Items relation, and saves them to a new relation Stuff and rename the attributes to Noms and NotNoms.

3.4 Cartesian Product

The *Cartesian Product* binary operator denoted $\times$ takes relations (S, R) and (S', R') and returns the relation $(S \cup S', R'')$ where R'' is a set of tuples $\{rr' | r \in R, r' \in R'\}$.

3.5 Natural Join

The *natural join* binary operator denoted $*$ takes relations (S, R) and (S', R') where $J = S \cap S' \neq \emptyset$, and returns $(S \cup S', R'')$ where R'' is the set of tuples $\{rr' | r \in R, r' \in R', r(J) = r'(J)\}$.

3.6 Join

The *join* operator denoted $\otimes$ takes relations (S, R), (S', R'), and a function f, and returns the relation $(S \cup S, R'')$ where R'' is the set of tuples $\{rr' | r \in R, r' \in R', f(r, r') = true\}$.

For example:

CALL $\otimes_{\mathrm{CALL.portid\ =\ CALL\_FORWARD\_NUMBERS.portid}}$ CALL_FORWARD_NUMBERS

Joins CALL and CALL_FORWARD_NUMBERS when CALL.portid = CALL_FORWARD_NUMBERS.portid.

3.7 Aggregate Function

The *Aggregate Function* takes a set of attributes from the relation, a function list, and a relation and returns the result of applying the function to the tuples of the relation and grouping by the given attributes. Available functions are SUM, AVERAGE, MINIMUM, MAXIMUM, COUNT.

For example:

lineid $\mathcal{F}_{\text{count scode}}$ SERVICE_SUBSCRIBERS

Returns the count of unique values for attribute scode, grouped by lineid on relation SERVICE_SUBSCRIBERS.

Appendix A

Entity Relational Diagrams

Entity Relational Diagrams are a visual diagramming language for describing entities using relational vocabulary.