**COS60009 – Database Management System for the Big Data age**

**Individual Assignment -2**

Name: Arun Ragavendhar Arunachalam Palaniyappan

Student ID: 104837257

**a) Consider the universal relation R = {A, B, C, D, E, F, G, H} and the set of functional dependencies *F* = {{A, B} →{C}, {A}→{D}, {E}→{F}, {D}→{G, H}}.**

#### 1) What is the key for R?

#### 2) If R is not in 2NF, first decompose it into 2NF and then 3NF relations; if R is in 2NF but not in 3NF, decompose it into 3NF relations; otherwise, do nothing.

**Solution:**

**1)**Given relation R= {A, B, C, D, E, F, G, H}

And the functional dependency between attributes

F= {A, B→C, A→D, E→F, D→G, H}

* {A, B} -> {C} (AB determines C)
* {A} -> {D} (A alone determines D)
* {E} -> {F} (E alone determines F)
* {D} -> {G, H} (D alone determines both G and H)

#### Candidate Key Analysis for each attribute and checking if they can derive all attributes:

* **A:** A+= {A, D, G, H}
* **B:** B+= {B, C}
* **C:** C+= {C}
* **D:** D+= {D, G, H}
* **E:** E+= {E, F}
* **F:** F+= {F}
* **G:** G+= {G}
* **H:** H+= {H}

None of the single attributes can derive all attributes, so no single attribute is a candidate key.

#### Combining the Attributes and checking if they can derive all attributes:

* **ABE:**
  + (AB)+= {A, B, C, D, G, H}
  + (E)+= {E, F}
  + (ABE)+= {A, B, C, D, E, F, G, H}

**Checking if any attribute can be removed from {A, B, E} while still maintaining the closure property:**

* **Removing A:**
  + (BE)+= {B, C, E, F} (Does not include all attributes)
* **Removing B:**
  + (AE)+= {A, C, D, E, F, G, H} (Does not Includes all attributes)
* **Removing E:**
  + (AB)+= {A, B, C, D, G, H} (Does not include all attributes)

**Hence the key is for R🡪 { A, B, E } .**

#### 2)Decomposition into 2NF and 3NF:

The relation is currently not in 2NF as there are partial dependencies.

##### **2NF Decomposition:**

Removing Partial dependencies:

* R1(A, B, C) from {A, B} →{C}
* R2(A, D) from {A}→{D}
* R3(E, F) from {E}→{F}
* R4(D, G, H) from {D} → {G, H}

Now, all relations are in 2NF.

To reduce to 3NF, we need to ensure there are no transitive dependencies.

##### 3NF Decomposition:

Removing transitive dependencies to reduce to 3NF.

* R1(B, C)
* R2(A, C)
* R3(A, D)
* R4(E, F)
* R5(D, G, H)

All relations are in 3NF now.

This completes the decomposition of Relation into 2NF and 3NF relations.

**b) Design a proper DTD for the FitnessCenter XML document.**

**Solution:**

<!DOCTYPE FitnessCenter

[

<!ELEMENT FitnessCenter (Member\*)>

<!ELEMENT Member (Name, Phone+, FavoriteColor)>

<!ELEMENT Name (#PCDATA)>

<!ELEMENT Phone (#PCDATA)>

<!ELEMENT FavoriteColor (#PCDATA)>

<!ATTLIST Member

id ID #REQUIRED

level CDATA #REQUIRED>

<!ATTLIST Phone

type (home|work) #REQUIRED>

]

>

**c) For the following queries on the XML document below for FitnessCenter, write the corresponding XPath expressions.**

**1) Find the home phone number of Jeff.**

**2) Find the id of all members at the platinum level.**

**3) When pointing to the level attribute of “David”, find the level attribute**

**of the member next to “David”.**

**Solution:**

The xpath queries:

1)./FitnessCenter/Member[Name='Jeff']/Phone[@type='home']

2). /FitnessCenter/Member[@level='platinum']/@id

3). /FitnessCenter/Member[Name='David']/@level/../following-sibling::Member/@level