CENG 352 - Database Management Systems
Written Assignment 1

Yavuz Selim YESILYURT
2259166

05.04.2020

1 E/R Diagram

a) Necessary SQL DDLs to resemble the E/R diagram are below:

```
CREATE TABLE Department (
dept_id int,
location varchar(50),
name varchar(50),
PRIMARY KEY (dept_id));
```

```
CREATE TABLE Employee (
emp_id int,
surname varchar(50),
name varchar(50),
salary int,
gender varchar(50),
PRIMARY KEY (emp_id));
```

```
CREATE TABLE Reports_to (
supervisor_emp_id int,
subordinate_emp_id int,
PRIMARY KEY (supervisor_emp_id, subordinate_emp_id),
FOREIGN KEY (supervisor_emp_id) REFERENCES Employee(emp_id),
FOREIGN KEY (subordinate_emp_id) REFERENCES Employee(emp_id));
```

```
CREATE TABLE Works_in (
dept_id int not null,
emp_id int,
PRIMARY KEY (dept_id, emp_id),
FOREIGN KEY (dept_id) REFERENCES Department(dept_id) ON DELETE NO AC-
TION,
FOREIGN KEY (emp_id) REFERENCES Employee(emp_id) ON DELETE CAS-
CADE);
```

```
CREATE TABLE Manages (
dept_id int,
emp_id int not null DEFAULT 101,
PRIMARY KEY (dept_id),
FOREIGN KEY (dept_id) REFERENCES Department(dept_id),
FOREIGN KEY (emp_id) REFERENCES Employee(emp_id) ON DELETE SET DE-
FAULT);
```

```
CREATE TABLE Runs_Project (
dept_id int,
project_id int,
budget int,
due_date date,
state varchar(50),
PRIMARY KEY (dept_id, project_id),
FOREIGN KEY (dept_id) REFERENCES Department(dept_id));
```

b) An assertion called Total to enforce the total participation constraint:

```
CREATE ASSERTION Total
CHECK (NOT EXISTS (
SELECT emp_id
FROM Employee
EXCEPT
SELECT DISTINCT emp_id
FROM works_in))
```

c) SQL CHECK clauses to implement the given constraints:

```
CHECK (Salary > 3600)
```

```
CHECK (name LIKE CONCAT('%', location, '%'))
```

d) An after trigger to implement the given constraint:

```
CREATE TRIGGER CheckBudgetOnChange
AFTER UPDATE OF budget ON Runs_Project
REFERENCING
OLD ROW AS OldRow
NEW ROW AS NewRow
FOR EACH ROW
WHEN (OldRow.budget > NewRow.budget)
UPDATE Runs_Project SET state = 'Unsuccessful'
WHERE Runs_Project.dept_id = OldRow.dept_id AND
Runs_Project.project_id = OldRow.project_id
```

2 Maximum Number of Tuples

a) Assume that the Product table consists of 100 rows with unique id's, the Store table has 5 rows with unique id's and the Person table contains 1000 rows with unique id's.

We have the following definition of relation R for the given E/R diagram:

$$\text{R}(\underline{\text{prodId}}, \underline{\text{storeId}}, \text{personId})$$

We will have a distinct record for each unique primary key tuple, i.e. a distinct record for each unique "prodId" and "storeId" attributes. Since there are 100 unique products in Product table and 5 stores in Store table, we will have $100 * 5 = 500$ records in relation R as a result.

b) Assume that the Product table consists of 100 rows with unique id's, the Store table has 5 rows with unique id's and among 1000 Person entries there are 10 SalesPerson.

We have the following definition of relation R for the given E/R diagram:

$$\text{R}(\underline{\text{prodId}}, \text{storeId}, \text{sales\_personId}, \underline{\text{customer\_personId}})$$

We will have a distinct record for each unique primary key tuple, i.e. a distinct record for each unique "prodId" and "customer_personId" attributes. Since there are 100 unique products in Product table and 990 customers in Person table, we will have $100 * 990 = 99000$ records in relation R as a result.

3 Armstrong's Axioms

First let us enumerate the given functional dependencies to refer them later in the proofs. We have:
1) A $\rightarrow$ C
2) B $\rightarrow$ E
3) CB $\rightarrow$ F
4) FE $\rightarrow$ G
5) FG $\rightarrow$ AH

I am going to continue enumerating also my findings to be able to refer them later.

a) let us prove CB $\rightarrow$ G by using Armstrong's Axioms

6) CB $\rightarrow$ B (Trivial)
7) CB $\rightarrow$ E (Transitivity on 2, 6)
8) CB $\rightarrow$ FE (Split/combine on 3, 7)
9) CB $\rightarrow$ G (Transitivity on 4, 8)
Therefore we have CB $\rightarrow$ G.

b) let us prove AB $\rightarrow$ EF by using Armstrong's Axioms

6) AB $\rightarrow$ B (Trivial)
7) AB $\rightarrow$ E (Transitivity on 2, 6)
8) AB $\rightarrow$ A (Trivial)
9) AB $\rightarrow$ C (Transitivity on 1, 8)
10) AB $\rightarrow$ CB (Trivial (on 9))
11) AB $\rightarrow$ F (Transitivity on 3, 10)
12) AB $\rightarrow$ EF (Split/combine on 7, 11)
Therefore we have AB $\rightarrow$ EF.

4 Normal Forms

a) To find all the keys of this relation we need to find closures of all the combinations of attributes in R as follows:

$A^+ = \{A, B\}$, $B^+ = \{B\}$, $C^+ = \{C\}$, $D^+ = \{C, D\}$, $E^+ = \{E, G\}$, $F^+ = \{D, F\}$, $G^+ = \{G\}$, $AB^+ = \{A, B\}$, $AC^+ = \{A, B, C\}$, $AD^+ = \{A, B, C, D\}$, $AE^+ = \{A, B, E, G\}$, $AF^+ = \{A, B, C, D, E, F\}$, $AG^+ = \{A, B, G\}$.
Since $AF^+ = \{A, B, C, D, E, F\}$ and AF^+ contains all elements, AF is key.
Continuing in this pattern with the two attribute dependencies on left-handside we'll end up with no other keys that have 2 columns.

b) Is R in BCNF or not?

In relation R we have the following Functional Dependency A $\rightarrow$ B which is both non-trivial and does not have a (super)key on its left-handside. If relation R were to in BCNF, attribute on the left-handside (A) would have been the key(AF) or a superkey. We see that it is not the case here, so R is not in BCNF.

c) If it is not in BCNF form decompose it into a collection of BCNF relations.

$$\begin{tikzcd}
 & \text{R(A,B,C,D,E,F,G)} \arrow[dl, no head] \arrow[dr, no head] & & & & \\
R_1\text{(A,B)} & & R_2\text{(A,C,D,E,F,G)} \arrow[dl, no head] \arrow[dr, no head] & & & \\
 & R_{21}\text{(C,D,E)} & & R_{22}\text{(A,D,F,G)} \arrow[dl, no head] \arrow[dr, no head] & & \\
 & & R_{221}\text{(F,D)} & & R_{222}\text{(A,F,G)} \arrow[dl, no head] \arrow[dr, no head] & \\
 & & & R_{2221}\text{(F,G)} & & R_{2222}\text{(A,F)}
\end{tikzcd}$$

d) Show that the above decomposition is,

- Dependency-preserving or not?

  We can easily deduct that if we join the final relations into one we won't get some certain dependencies back in the global combined relation. For example we see $E \rightarrow G$ and $AC \rightarrow D$ were in the initial global functional dependency list and we've lost them during the decomposition. Therefore, this decomposition is not dependency-preserving.

- Lossless-join or not?

  We know that any BCNF decomposition is always lossless and here in above we did a BCNF decomposition; but in more detail, we can see that we are able to join final (decomposed) relations using at least one's primary key field and whenever we are able to achieve such joins on all relations we won't have any problems with losing data in the global joined relation.

5 Normalization

a) List of all FDs you identified and the corresponding SQL queries to discover them at the end of step 3 above.

$A \rightarrow E$

```
SELECT a
FROM q5_table
GROUP BY a
HAVING COUNT(DISTINCT e) > 1
```

$AB \rightarrow C$

```
SELECT a
FROM q5_table
GROUP BY a, b
HAVING COUNT(DISTINCT c) > 1
```

$AB \rightarrow E$

```
SELECT a
FROM q5_table
GROUP BY a, b
HAVING COUNT(DISTINCT e) > 1
```

$AC \rightarrow B$

```
SELECT a
FROM q5_table
GROUP BY a, c
HAVING COUNT(DISTINCT b) > 1
```

$AC \rightarrow E$

```
SELECT a
FROM q5_table
```

```
GROUP BY a, c
HAVING COUNT(DISTINCT e) > 1
```

$AD \to E$

```
SELECT a
FROM q5_table
GROUP BY a, d
HAVING COUNT(DISTINCT e) > 1
```

$BC \to A$

```
SELECT b
FROM q5_table
GROUP BY b, c
HAVING COUNT(DISTINCT a) > 1
```

$BC \to E$

```
SELECT b
FROM q5_table
GROUP BY b, c
HAVING COUNT(DISTINCT e) > 1
```

$BE \to A$

```
SELECT b
FROM q5_table
GROUP BY b, e
HAVING COUNT(DISTINCT a) > 1
```

$BE \to C$

```
SELECT b
FROM q5_table
GROUP BY b, e
HAVING COUNT(DISTINCT c) > 1
```

$C \to A$

```
SELECT c
FROM q5_table
GROUP BY c
HAVING COUNT(DISTINCT a) > 1
```

$C \to B$

```
SELECT c
FROM q5_table
GROUP BY b
HAVING COUNT(DISTINCT b) > 1
```

$C \to E$

```
SELECT c
FROM q5_table
GROUP BY e
HAVING COUNT(DISTINCT e) > 1
```

$CD \to A$

```
SELECT c
FROM q5_table
GROUP BY c, d
HAVING COUNT(DISTINCT a) > 1
```

$CD \to B$

```
SELECT c
FROM q5_table
GROUP BY c, d
HAVING COUNT(DISTINCT b) > 1
```

$CD \to E$

```
SELECT c
FROM q5_table
GROUP BY c, d
HAVING COUNT(DISTINCT e) > 1
```

$CE \to A$

```
SELECT c
FROM q5_table
GROUP BY c, e
HAVING COUNT(DISTINCT a) > 1
```

$CD \to B$

```
SELECT c
FROM q5_table
GROUP BY c, d
HAVING COUNT(DISTINCT b) > 1
```

$DE \to A$

```
SELECT d
FROM q5_table
GROUP BY d, e
HAVING COUNT(DISTINCT a) > 1
```

$E \rightarrow A$

```
SELECT e
FROM q5_table
GROUP BY e
HAVING COUNT(DISTINCT a) > 1
```

b) List of all SQL statements to create normalized tables.

```
CREATE TABLE q5_AE (
A varchar(20),
E varchar(20),
PRIMARY KEY(A)
);

CREATE TABLE q5_ABC (
A varchar(20),
B varchar(20),
C int,
PRIMARY KEY(C),
FOREIGN KEY(A) REFERENCES q5_AE(A)
);

CREATE TABLE q5_CD (
C int,
D int,
PRIMARY KEY(C,D)
);
```

c) List of all SQL statements that load the contents of the tables.

```
INSERT INTO q5_AE(a,e)
SELECT DISTINCT a,e
FROM q5_table

INSERT INTO q5_ABC(a,b,c)
SELECT DISTINCT a,b,c
FROM q5_table

INSERT INTO q5_CD(c,d)
SELECT DISTINCT c,d
```

```
FROM q5_table
```