NYU Tandon School of Engineering **Nov 16, 2017**
Computer Science and Engineering
CS 6083, Fall 2017

Problem Set # 3 Sample Solution

Problem2.

(a) Candidate key: {BFG}

(b) Canonical cover: Fc = {BF → C, CF → D, G → AE, F G → D}.

(c) It's not in BCNF because there is nontrivial functional dependency BF → C, and BF is not superkey.
Decomposing relation R into BCNF:
Because BF → C : R1 = (B, C, F), and R2 = (A, B, D, E, F, G).
Because G → AE, and G is not superkey, decomposing relation R2 into BCNF:
R2 = (A, E, G), and R3 = (B, D, F, G).
Because F G → D, and F G is not superkey, decomposing relation R3 into BCNF:
R3 = (D, F, G), and R4 = (B, F, G).

And the BCNF form is: R1 = (B, C, F), R2 = (A, E, G), R3 = (D, F, G), and R4 = (B, F, G).

(d) No. The BCNF form in (c) is not dependency preserving. We cannot check CF → D in the result of (c).
3NF form: R1 = (B, C, F), R2 = (C, D, F), R3 = (A, E, G), R4 = (D, F, G), and R5 = (B, F, G).

Problem3.

(a) This is not a good design for following reasons:

1) Current schema does not reflect a lot of functional dependencies. For instance, as assumptions in the question, {pid → pname, page, pstate} holds, but since the {pid} is not a candidate key, we will end up repeating this information in several tuples.

2) Inserting values for the separate entities in one table like Patient or Doctor will lead to storing NULL and duplicate values at several places. This will increase the storage and our data will be inconsistent.

3) We have to access this one table for querying even minimal data, this will be time consuming as everything is under this table.

4) Lastly, this schema does not adhere to the normalization rules. As everything is under one table, it will make maintenance and querying from the database difficult and it will be hard to see relation between each attribute. it's better to divide this big table to many smaller ones.

(b) Candidate Keys: {pid, docid, did, rid, ttime}

(c)

{pid} → {pname, page, pstate}
{docid} → {docname, docage, doclevel,docsalary}

{did} → {dname, dtype, description}
{rid}→ {rname, rcapacity}
{doclevel} → {docsalary}
{rid}→{dtype}
{pid,did,docid,ttime} → {tcost, insurancediscount}

(d) the canonical cover for the functional dependencies in F is:
{pid} → {pname, page, pstate}
{docid} → {docname, docage, doclevel}
{doclevel} → {docsalary}
{did} → {dname, dtype, description}
{rid}→ {rname, rcapacity,dtype}
{pid,did,docid,ttime} → {tcost, insurancediscount}

(e) No, it is not in BCNF. Because for schema to be BCNF for each non-trivial dependency A->B, A should be a superkey, which is not the case here.
Convert into BCNF form:

Step 1: Relation PatientTreatment can be decomposed into PATIENT and R2 according to the violation by dependency {pid} → {pname, page, pstate}, we can get relation Patient:
Patient (pid, pname, page, pstate)
R2= {pid, docid, docname, docage, doclevel, docsalary, did, dname, dtype, description, rid, rname, rcapacity, ttime, tcost, insurancediscount}

Step 2: According to the violation by functional dependency {docid} → {docname, docage, doclevel}, We can get relation Doctor:
Doctor (docid, docname, docage, doclevel)
R3={pid, docid, docsalary, did, dname, dtype, description, rid, rname, rcapacity, ttime, tcost, insurancediscount}

Step3: According to the violation by functional dependency {doclevel} → {docsalary}, We can get relation Salary:
Salary (doclevel, docsalary)
R4={pid, docid,did, dname, dtype, description, rid, rname, rcapacity, ttime, tcost, insurancediscount}

Step4: According to the violation by functional dependency {did} → {dname, dtype, description}, We can get relationDisease:
Disease (did, dname, dtype, description)
R5 ={pid, docid, did, rid, rname, rcapacity, ttime, tcost, insurancediscount}

Step 5: According to the violation by functional dependency, {rid}→ {rname, rcapacity, ,dtype}, we can get relation Room:
Room (rid, rname, rcapacity)
R6 ={pid, docid,did, rid, ttime, tcost, insurancediscount}

Step 6: According to the violation by functional dependency {pid,did,docid,ttime} → {tcost, insurancediscount, we can get relation Cost:
Cost (pid, did, docid, ttime, tcost, insurancediscount)
R7 ={pid, docid, did, rid, ttime}

Thus, the BCNF form is:
Patient (pid, pname, page, pstate)
Doctor (docid, docname, docage, doclevel)
Salary (doclevel, docsalary)

Disease (did, dname, dtype, description)
Room (rid, rname, rcapacity)
Cost (pid, did, docid, ttime, tcost, insurancediscount)
PatientTreatment (pid, docid, did, rid, ttime)

(f) No, the schema in section(e) is not dependency preserving because the functional dependency {rid}→ {dtype} can not be checked in the BCNF form. Thus, the 3NF form is:

Patient (pid, pname, page, pstate)
Doctor (docid, docname, docage, doclevel)
Salary (doclevel, docsalary)
Disease (did, dname, dtype, description)
Room (rid, rname, rcapacity,dtype)
Cost (pid, did, docid, ttime, tcost, insurancediscount)
PatientTreatment (pid, docid, did, rid, ttime)

(g) The candidate key is {pid, docid, did, rid, ttime}

Functional dependency:
{pid} → {pname, page, pstate}
{docid} → {docname, docage, doclevel,docsalary}
{did} → {dname, dtype, description}
{rid}→ {rname, rcapacity}
{doclevel} → {docsalary}
{rid}→{dtype}
{pid,did,docid,ttime} → {tcost}
{pstate,tcost} → {insurancediscount}

Canonical Cover:
{pid} → {pname, page, pstate}
{docid} → {docname, docage, doclevel}
{did} → {dname, dtype, description}
{rid}→ {rname, rcapacity,dtype}
{doclevel} → {docsalary}

{pid,did,docid,ttime} → {tcost}
{pstate,tcost} → {insurancediscount}

No, it's not in BCNF form, convert it into BCNF:
Patient (pid, pname, page, pstate)
Doctor (docid, docname, docage, doclevel)
Salary (doclevel, docsalary)

Disease (did, dname, dtype, description)
Room (rid, rname, rcapacity)
Cost (pid, did, docid, ttime, tcost, insurancediscount)
PatientTreatment (pid, docid, did, rid, ttime)

No, the schema in section(e) is not dependency preserving because the functional dependency {rid}→ {dtype} can not be checked in the BCNF form. Thus, the 3NF form is:

Patient (pid, pname, page, pstate)
Doctor (docid, docname, docage, doclevel)
Salary (doclevel, docsalary)
Disease (did, dname, dtype, description)
Room (rid, rname, rcapacity,dtype)
Cost (pid, did, docid, ttime, tcost)
Insurance (pstate, tcost, insurancediscount)
PatientTreatment (pid, docid, did, rid, ttime)

Problem 4.
Use MySQL
(a)

```
SELECT TABLE_NAME, COLUMN_NAME
FROM INFORMATION_SCHEMA.TABLES natural join INFORMATION_SCHEMA.COLUMNS
WHERE COLUMN_KEY = 'PRI'
AND TABLE_SCHEMA = 'hw1'
```

```
Select TABLE_NAME, COLUMN_NAME
From information_schema.TABLES natural join information_schema.COLUMNS
Where TABLE_SCHEMA ='hw1' AND COLUMN_KEY ='PRI'
```

100% 1:1

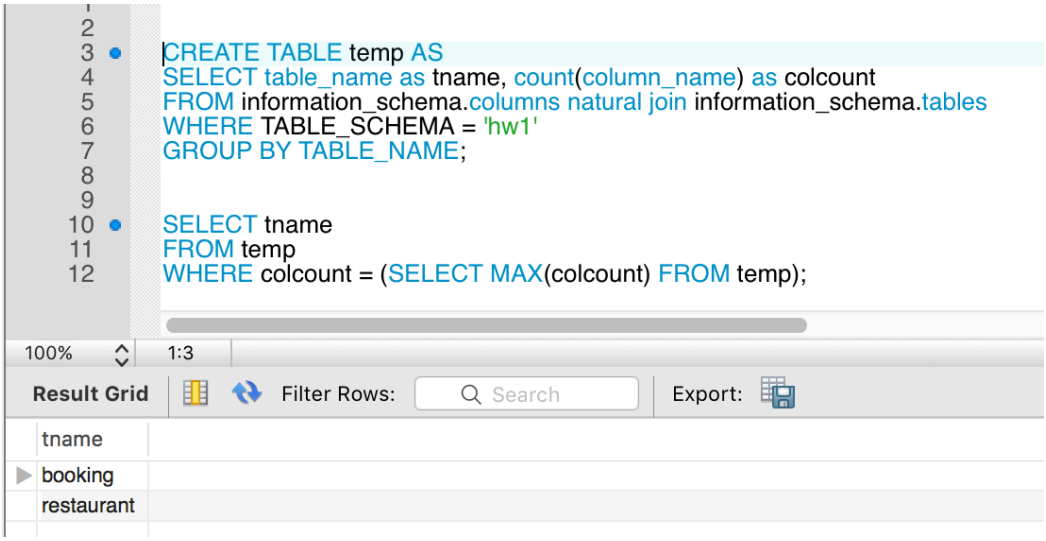
Result Grid | Filter Rows: Search | Export:

TABLE_NAME	COLUMN_NAME
booking	bid
customer	cid
restaurant	rid

(b)

```
CREATE TABLE temp AS
SELECT table_name as tname, count(column_name) as colcount
FROM information_schema.columns natural join information_schema.tables
WHERE TABLE_SCHEMA = 'hw1'
GROUP BY TABLE_NAME;


SELECT tname
FROM temp
WHERE colcount = (SELECT MAX(colcount) FROM temp);
```



(c)

```
SELECT table_name,column_name
FROM information_schema.columns natural join information_schema.tables
WHERE table_schema = 'hw1' and DATA_TYPE = 'int';
```

```
SELECT table_name,column_name
FROM information_schema.columns natural join information_schema.tables
WHERE table_schema = 'hw1' and DATA_TYPE = 'int';
```

100% 1:1

Result Grid | Filter Rows: Search | Export:

table_name	column_name
booking	bid
booking	cid
booking	rid
booking	quantity
customer	cid
restaurant	rid
restaurant	capacity

(d)

```
CREATE TABLE temp2 AS
SELECT table_name, column_name, data_type
FROM information_schema.columns natural join information_schema.tables
where table_schema = 'hw1';
SELECT distinct t1.table_name, t2.table_name
FROM temp2 as t1, temp2 as t2
WHERE t1.table_name < t2.table_name and t1.column_name = t2.column_name and
t1.data_type = t2.data_type;
```

100% 1:1

Result Grid | Filter Rows: Search | Export:

table_name	table_name
booking	customer
booking	restaurant