CS220P: Databases and Data Management – Fall 2022

Homework 3: Relational Database Design Theory (100 points)

Due Date: **Thursday, October 20 (11:59 PM)**

Student Name: Jiaxin Li

Student ID: 19683688

1. [10 pts]

Review the ER diagram provided as a solution to HW1 and the SQL DDL provided as a solution to HW2 - review them carefully! List any business rules called out in HW1's description that are not captured in this current SQL DDL. For each, indicate whether it is **(i)** reflected in the ER diagram but not in the SQL DDL or **(ii)** missing in both the ER diagram and the SQL DDL. An example answer in your list might be "The rule that a User must not be a cat is missing in both the ER diagram and the SQL DDL." (That is, if the initial problem statement had said that.) [10 points]

1. The derived attribute rating for Seller only appears in ER diagram not SQL DDL.
2. The Ad plan level can only have 4 values (bronze, silver, gold, plantinum) which is missed in both ER and SQL DDL.
3. The rating from buyer for seller for quality, pricing, delivery and rating date are one-to-five star scale. This restriction on rating is missed in both ER and SQL DDL.
4. The rule that it is required for a user to be buyer or seller or both is missed in SQL DDL but not missed in ER diagram.
5. The rule that item is either good or service is in ER but not in SQL DDL

2. [20 pts]

To expand our business, we have been asked to add location-based ads. Each Ad should be pushed to areas where the seller wants to (or can) provide the Goods or Services. Users can specify their city and state of interest and get to see all the items available to them. Thus, city and state information needs to be added to the Ad table to indicate where an item will be available. Based on their understanding of these requirements, one of our old-school database designers -- one of those "go straight to the tables" folks -- has proposed the following design to **replace** the current Interchange.com Ad table.

```
CREATE TABLE LocationBasedAd(
   ad_id          text NOT NULL,
   plan           text NOT NULL ,
   content        text,
   pic_num        int NOT NULL,
   item_id        text NOT NULL,
   seller_user_id  text,
   placed_date     date NOT NULL,
   city           text NOT NULL,
   state          text NOT NULL,
   PRIMARY KEY (ad_id, city, state),
   FOREIGN KEY(item_id) REFERENCES Item(item_id) ON DELETE CASCADE,
   FOREIGN KEY(pic_num, item_id) REFERENCES Picture(pic_num, item_id) ON DELETE CASCADE,
   FOREIGN KEY(seller_user_id) REFERENCES Seller(user_id) ON DELETE CASCADE
);
```

(a) [5 pts] Looking at the primary key constraint of this table, what non-trivial functional dependency (or dependencies) can you infer, if any?

ad_id, city, state → plan, content, pic_num, item_id, seller_user_id, placed date

(b) [5 pts] An intern on the team -- a CS122A survivor -- says that this design doesn't make sense to her after looking at the proposed CREATE TABLE statement. When you ask her why, she vaguely says that it has several "key problems", but she doesn't want to say more for fear of offending the design's proposer. Your job is to correct the above design by fixing its constraints, e.g., by offering a corrected PRIMARY KEY clause.

PRIMARY KEY(ad_id)

(c) [5 pts] What non-trivial functional dependencies do your revised table have?

ad_id→plan, content, pic_num, item_id, seller_user_id, placed_date, city, state
item_id → seller_user_id

(d) [5 pts] If LocationBasedAd is not already at least in 3NF, then normalize it into at least 3NF and show the resulting relation(s) and specify their candidate keys. Make sure that your 3NF decomposition is both lossless-join and dependency-preserving. Note: If LocationBasedAd was already in 3NF, then just list the candidate keys of LocationBasedAd. What is the highest normal form that your answer now satisfies?

LocationBasedAd(ad_id, plan, content, pic_num, item_id, placed_date, city, state)
Candidate key: ad_id

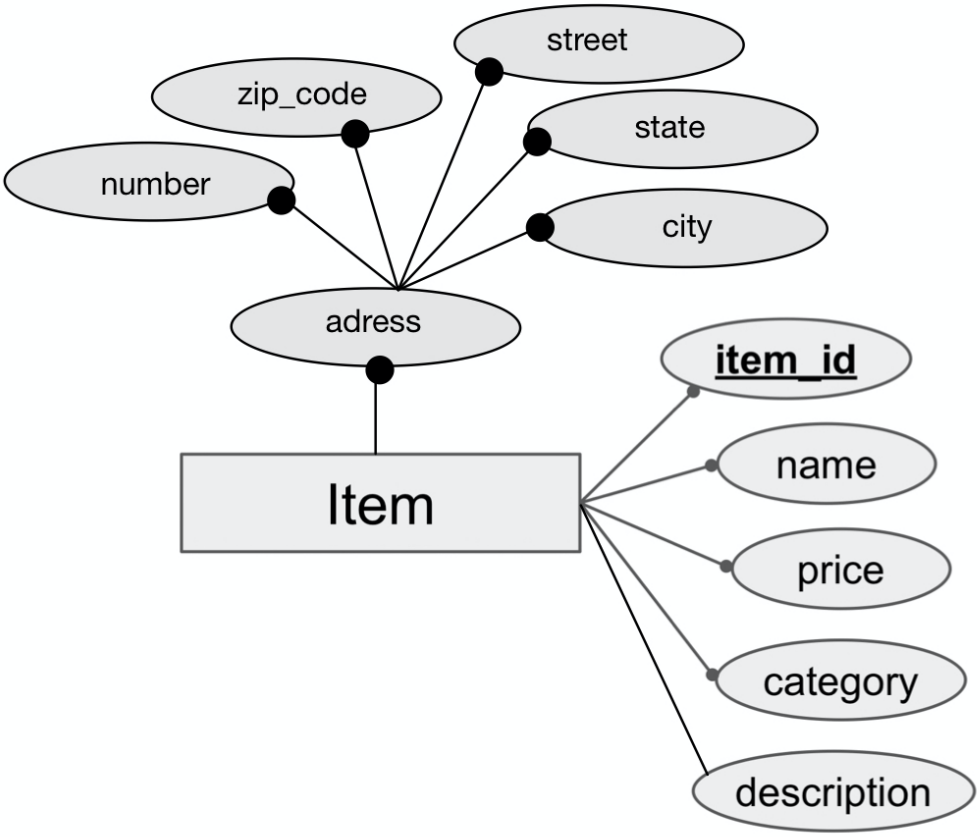
IS(item_id, seller_user_id)
Candidate key: item_id

BCNF

3. [20 pts]

Interchange.com is getting popular and your boss has been getting requests to allow users to specify where items would be delivered (buyers have been ordering stuff for their friends and families). Your boss at Interchange.com wants to add delivery addresses for each sold item in the Item table. You realize that a physical address is thus needed for sold Items in addition to the buyer's id and purchase date in the Item table. The physical address should include the usual: number, street, city, state, and zip code. **Assume that a zip code can include multiple cities but not multiple states[1].**

(a) [4 pts] Draw a revised E-R diagram for the Item entity and any affected relationships. Your diagram should only show the Item entity set and any affected relationship sets with all of their attributes. You may draw your diagram with any software you prefer, take a screenshot, and paste it below.



[1] This is not true in the real world, but you are to assume it here. (Don't argue with your boss. :-))

(b) [4 pts] Provide appropriate table creation DDL for the revised Item table (which we will be referring to as Item for the rest of this problem):

```
CREATE TABLE Item(
    item_id          text NOT NULL,
    name             text NOT NULL ,
    price            text NOT NULL,
    category         text NOT NULL,
    description      text,
    buyer_id         text,
    seller_id        text NOT NULL,
    buyer_id         text NOT NULL,
    sold_date        date NOT NULL,
    list_date        date NOT NULL,
    address_number   text NOT NULL,
    address_zipcode  text NOT NULL,
    address_street   text NOT NULL,
    address_city     text NOT NULL,
    address_state    text NOT NULL,
    PRIMARY KEY (item_id),
    FOREIGN KEY(buyer_id) REFERENCES seller(user_id) ON DELETE CASCADE,
    FOREIGN KEY(seller_id) REFERENCES seller(user_id) ON DELETE CASCADE,
);
```

(c) [4 pts] What are the functional dependencies that hold on your Item table?

item_id → name, price, category, description, buyer_id, sold_date, list_date, address_number, address_zipcode, address_street, address_city, address_state

address_zipcode→address_state

(d) [4 pts] Normalize the Item table into 3NF, show the resulting relations, and specify their candidate keys. Make sure that your 3NF decomposition is both lossless-join and dependency-preserving. What is the highest normal form that your answer now satisfies?

Item(item_id, name, price, price, category, description, sold_date, buyer_id, address_number, address_zipcode, address_street, address_city)

Candidate Key: item_id

ZipState(address_zipcode, address_state)

Candidate Key: address_zipcode

Both Satisify BCNF

(e) [4 pts] Take a thoughtful look at the Item table in the resulting decomposed tables and what they will mean for the Interchange.com application and query team. Is it definitely a good idea to use these tables instead of the one in part (b)? Why or why not? (Identify any tradeoff(s) you see.)

No, because it increases the cost of query when wanting to get the adress_state from an item. It indeed reducing some redundancy but is limited and is not worthwhile when considering the cost of query it brings.

4. [15 pts]

Consider the following relation:

F	**E**	**D**
f_5	e_2	d_4
f_3	e_3	d_1
f_2	e_5	d_2
f_4	e_2	d_2
f_4	e_1	d_3
f_1	e_4	d_4

(a) [10 pts] Given the current state of the database, and for each one of the following functional dependencies, answer: a) Does this functional dependency hold in the above relation instance [Yes/No]? b) If your answer to the previous question was "No", explain why by listing a tuple that causes a violation.

i) $F \rightarrow E$

No: (f_4, e_2, d_2), (f_4, e_1, d_3)

ii) $E \rightarrow D$

No: (f_5, e_2, d_4), (f_4, e_2, d_2)

iii) $D \rightarrow E$

No: (f_4, e_2, d_2), (f_2, e_5, d_2)

iv) $E \rightarrow F$

No: (f_4, e_2, d_2), (f_5, e_2, d_4)

v) $F \rightarrow D$

No: (f_4, e_2, d_2), (f_4, e_1, d_3)

(b) [5 pts] List all ***potential*** candidate keys (if there are any) for the above relation that can be inferred from the given relation instance.

(FE), (FD), (ED)

Normalizing a schema with a set of FDs can be done automatically by computers. Complete questions 5 and 6 with the help of the normalization tool provided by Griffith University - *but try each part by hand first*! Use the problems to cement your understanding and use the tool to check your answers.

5. [20 pts]

Meeting (meeting_id(M), passcode(P), course_id(C), instructor_id(I), meeting_name(N), recurr_id (R))
(All attributes contain only atomic values.)

FD1: MC → I

FD2: M → IN

FD3: C → I

FD4: I → R

(a) [5 pts] Compute MC+, the attribute closure of the attribute set (meeting_id, course_id). Show each step of your work as well as your final result.

M → IN imply M→I and M→N

C → I and I →R imply C→R by transitivity

MC+ ={MCINR}

(b) [5 pts] List the candidate keys of Meeting. Is this related to your answer to (a)? Why?

MCP

Yes

According to (a), MC determine everything except P, so candidate keys are MCP.

(c) [5 pts] What's the highest normal form that Meeting satisfies and **why**?

1NF, because FD3 shows that there is an attribute partially dependent on the candidate key.

(d) [5 pts] If Meeting is not already at least in 3NF, then normalize Meeting into 3NF and show the resulting relation(s) and specify their candidate keys. Make sure that your 3NF decomposition is both lossless-join and dependency-preserving. Note: If Meeting was already in 3NF, then just list the candidate keys of R. What is the highest normal form that your answer now satisfies?

R0(MIN), candidate key: M

R1(CI), candidate key: C

R2(IR), candidate key: I

R3(MPC), candidate key: M,P,C

It satisfies BCNF

6. [15 pts]

Ad(ad_id(A), plan (P), content (C), pic_num(N), item_id(I), seller_user_id(S), placed_date(D))

(All attributes contain only atomic values.)

FD1: A → P

FD2: A → ISC

FD3: I → S

FD4: S → C

(a) [5 pts] Compute A+, the attribute closure of attribute A. Show your work and final result.

A→ISC imply A→I, A→S, A→C

A+ ={APISC}

(b) [3 pts] Is A → C in FD+, the closure of the set of FDs? Why or why not?

Yes, because C is in A+

(c) [4 pts] Normalize R into BCNF and show the resulting relation(s) and their candidate keys.

R0(API), candidate key: A

R1(IS), candidate key: I

R2(SC), candidate key:S

R3(AND), candidate key: A,N,D

(d) [3 pts] Is the decomposition in part (c) dependency-preserving? Why or why not?

Yes, all original F1,2,3,4 are included in part (c) decomposition.