INFSCI 2710 Database Management, Fall 2022

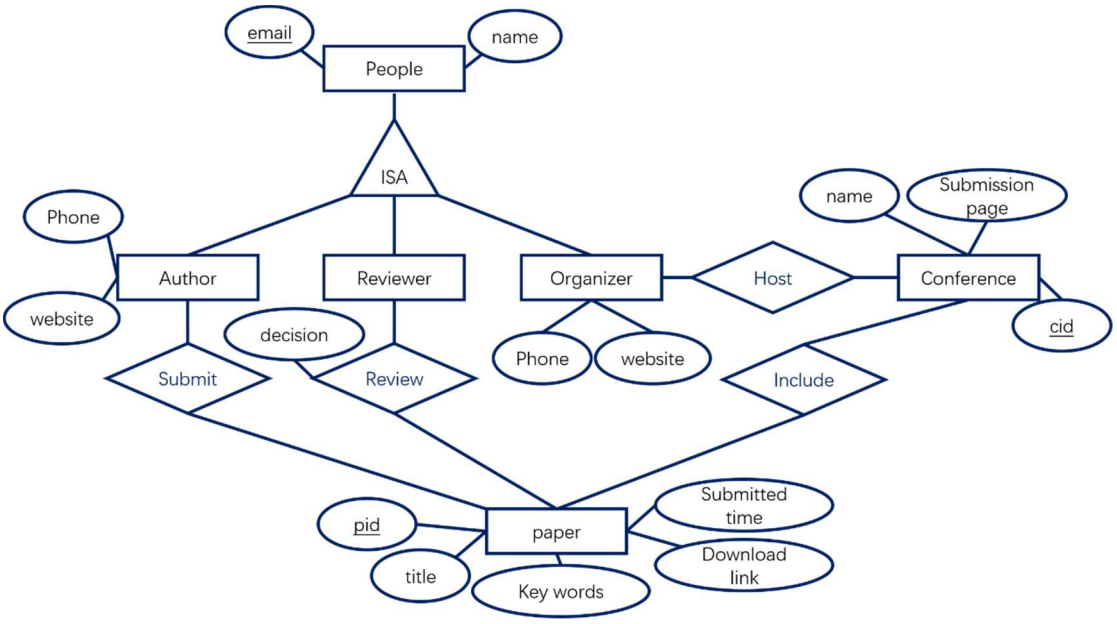
Homework 3: ER, Schema Refinement, Storage and Indexing (100 pts)

100 pts

Due Date: 11/17. Please submit a pdf to the Canvas assignment.

Q1 [10 pt] Draw an ER diagram for an online conference peer review system on the following description: The database must store information about people involved such as author who submits the paper, reviewer who reviews the paper and organizer who host the conference, also information about the conferences that papers are submitted to.

- Authors, reviewers, and organizers share some common attributes such as email (you can assume email as the unique identifier), and name, authors and organizer will have specific attributes: phone number, website, while reviewers will give a decision when review a paper. (You should use the ISA structure to describe the hierarchical structure)
- Each paper is described by id, title, authors, reviewers, conference, submitted time, key words and download link, one paper may be written by more than one author and reviewed by more than one reviewer, but each paper can only be published at one conference.
- Each conference should have information about its id, name, submission page, and organizers.
- You can draw ER diagram by hand, but make sure it's readable



Q2[10 pts] Translate the ER diagram from Q1 into SQL DDL statements

```
CREATE TABLE Author (
email VARCHAR(45) NOT NULL,
name VARCHAR(45) NOT NULL,
Phone VARCHAR(45) NOT NULL,
website VARCHAR(45) NOT NULL,
PRIMARY KEY (email)
);
```

```
CREATE TABLE Reviewer (
email VARCHAR(45) NOT NULL,
name VARCHAR(45) NOT NULL,
PRIMARY KEY (email)
);
```

```
CREATE TABLE Organizer (
email VARCHAR(45) NOT NULL,
name VARCHAR(45) NOT NULL,
Phone VARCHAR(45) NOT NULL,
website VARCHAR(45) NOT NULL,
PRIMARY KEY (email)
);

CREATE TABLE Paper (
pid VARCHAR(45) NOT NULL,
tittle VARCHAR(45) NOT NULL,
Submitted_Time DATE NOT NULL,
Key_words VARCHAR(45) NOT NULL,
Download_link VARCHAR(45) NOT NULL,
PRIMARY KEY(pId)
);

CREATE TABLE Conference (
cId VARCHAR(45) NOT NULL,
name VARCHAR(45) NOT NULL,
submission_page VARCHAR(45) NOT NULL,
PRIMARY KEY(cId)
);

CREATE TABLE Submit (
Email VARCHAR(45) NOT NULL,
pId VARCHAR(45) NOT NULL,
PRIMARY KEY(Email, PaperId ),
FOREIGN KEY(Email) REFERENCES Author(Email),
FOREIGN KEY(pId) REFERENCES Paper(pId)
);

CREATE TABLE Review (
Email VARCHAR(45) NOT NULL,
pId VARCHAR(45) NOT NULL,
Decision TEXT NOT NULL,
PRIMARY KEY(Email, pId ),
FOREIGN KEY(Email) REFERENCES Reviewer(Email),
FOREIGN KEY(pId) REFERENCES Paper(pId)
);

CREATE TABLE Host (
Email VARCHAR(45) NOT NULL,
cId VARCHAR(45) NOT NULL,
PRIMARY KEY(Email, cId),
FOREIGN KEY(Email) REFERENCES Organizer(Email),
FOREIGN KEY(cId) REFERENCES Conference(cId)
);
```

```
CREATE TABLE Include (
cId VARCHAR(45) NOT NULL,
pId VARCHAR(45) NOT NULL,
PRIMARY KEY(cId, pId),
FOREIGN KEY(cId) REFERENCES Conference(cId),
FOREIGN KEY(pId) REFERENCES Paper(pId)
);
```

Q3 [15 pts] Consider a relation R in table (a). Complete the table (b) for given functional dependencies (FD). Please just answer yes, no or unknown

A	B	C	D
A_1	B_1	C_1	D_1
A_2	B_1	C_2	D_2
A_1	B_2	C_1	D_1
A_3	B_1	C_2	D_2
A_4	B_3	C_3	D_3
A_4	B_1	C_1	D_1
A_5	B_4	C_3	D_1
A_5	B_3	C_2	D_3
A_3	B_4	C_3	D_1

Table (a)

FD	**Satisfied by R (yes/ no/unknown)**	**Hold on R (yes/no/unknown)**	**Trivial (yes/no)**
A→ B	N	N	N
B→ A	N	N	N
AC→ D	Y	Unknown	N
ABD→ B	Y	Y	Y
AC→B	N	N	N
AD→ B	N	N	N
C→ ABC	N	N	N
BC→ D	Y	Unknown	N
BD→ D	Y	Y	Y
BD→ A	N	N	N

Table (b)

- **Since the table only shows sample records from the relation R, even if FD is satisfied by all the instances shown, it is not necessarily hold in the relation R. But if FD is not satisfied in some of the records, then the FD Is definitely not hold in R**
- **A FD is trivial only if the attributes on the right-hand side is a subset of attributes on the left-hand side**

Q4 [10 pts] Consider a relation R1(A,B,C,D,E,F,G) and a set of functional dependencies FD = {AC → E, B → DE, F→ C, CD → GF} which hold on R1. Using Armstrong's axioms verify if the following functional dependencies hold on R1

FD	Yes/No	Proof if yes
ABC → G	Y	Given AC → E and B → DE, we have ABC → DE by union, then ABC → CDE by reflectivity. CD → GF is given, so we have ABC → CDEFG by transitivity, so ABC → G
AC→ F	N	
BF→G	Y	Because B→ DE, F→ C (given), BF→DECF(by union), also find CD→GF(given), so BF→GFE(transitivity), so BF→G
BCD→ F	Y	Because CD→GF, so CD→F, BCD→F
ABC→DE F	Y	Because AC→E, B→DE(given), so ABC→CDE (union), and because we have CD→GF(given), so ABC→DEF (union)

Q5 [10 pts] Consider a relation R(A,B,C,D,E,F) and a set of functional dependencies, which hold on R : { AB→E, C→BF, D→A } Are decompositions in the table lossless and why?

Decomposition	Lossless? (Yes) /(No)	Why
R1(ABF) and R2(CDE)	N	No common attribute
R1(ABCEF) and R2(CDE)	N	CE is common attributes and CE is not a key for either R1 or R2
R1(ABDE) and R2(ACDF)	N	AD is common attributes and AD is not a key for either R1 or R2
R1(ACDF) and R2(BCDE)	Y	CD is common attributes, and CD is a super key for R1

R1(ABEF) and R2(BCDF)	N	BF is common attribute. BF is not a key for either R1 or R2. thus, it is lossy decomposition

Q6 [10 pts] Consider the following relations with the associated functional dependencies. Decide, whether those relations are in (a) BCNF, (b)3NF, (c) neither in BCNF nor 3NF normal form.

Relation, FD	Answer (a, b, or c)	Solution
R1(A,B,C,D) {AB→ C, C→ D}	C	AB → D by transitivity, thus AB → ABCD by union and augmentation. AB is a key for R1. AB → C is trivial. C is not a key, D – C = D, and D is not a part of a key. Thus R1 is neither in BCNF nor in 3NF
R2(A,B,C,D), {AC→ BD}	A	AC → ABCD by augmentation, thus AD is a key. Thus, R2 is in BCNF
R3(A,B,C,D) {AB→ CD, D→A}	B	AB → ABCD by augmentation. Thus, AB is a key. D is not a key, B – D = B is a part of a key. Thus, R4 is in 3NF
R4(A,B,C,D,E), {AC→D, D→B}	C	Given AC →D and D→B, we have AC →ABCD, and ACE is the key by augmentation, so none of the FD has the key on the left side nor part of key on the right side. Thus, R4 is neither in BCNF nor in 3NF
R5(A,B,C,D,E) {A→ CE, D→ CE}	C	ABD → ABCDE by augmentation, thus ABD is a key. A is not a key, D is not a key. CE is not a part of the key. Thus, R5 is neither in BCNF nor in 3NF

NOTE: For the next questions, Q7 – Q10, you are running a DBMS on a computer that has 5 kByte disk block size. Reminder: 1kByte=1024bytes, 1 MByte = 1024 kBytes.

Q7 [5 pts] Table T in your database D has size 3 kBytes. How much space does Table T take on the drive? Show your calculations.

2 kBytes will takes one blocks in the disk, totally 5kBytes.

Q8 [5 pts] Table T in your database D has size 300 MBytes. You execute a query: "select * from T". How much data will be read from the drive? Assume that n*size_of_tuple = block_size, where n is a natural number. Show your calculations.

The assumption implies that every block on the disc which stores tuples from table T is full, there is no any empty space in blocks except of possibly the last one. Ceil (300*1024/5) = 61440 blocks. Thus, on the disk table T occupies 61440*5 = 300 Mbytes

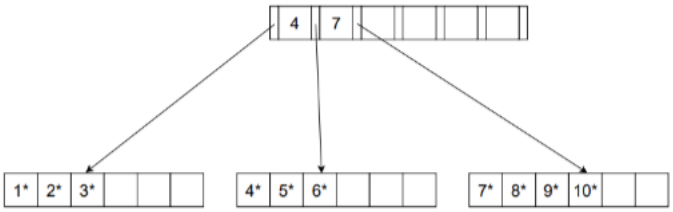
Q9 [5 pts] Assume that table T is defined in the same way as in question 8. You execute a query "select * from T where num=500". How many blocks and how many bytes will be read from the disk? Show your calculations (There is no index built on this column).

Because there is no index defined on table T that query will require to read the whole table T. Reusing result from Q8, it will be necessary to read 300MBytes for this query

Q10 [10pts] How your answer will be different from question 9, if clustered index for column num is used for table T. Assume that size of the required index structure is 8 kBytes and there are m records with num = 500, which may be stored in n different blocks.

the index is stored in ceil (8/5) = 2 blocks, plus the number of blacks that contain target records, which is 10 + 5*number of target blocks = 10 + 5n.

Q11 [10pts] Draw a valid B+ tree below for the search keys (1, 2, 3, 4, …, 10). Assume the keys are inserted in their natural order. The order of the tree is 3


4
7
1*
2*
3*
4*
5*
6*
7*
8*
9*
10*