8 HOUR ASSESSMENT FOR IS5102

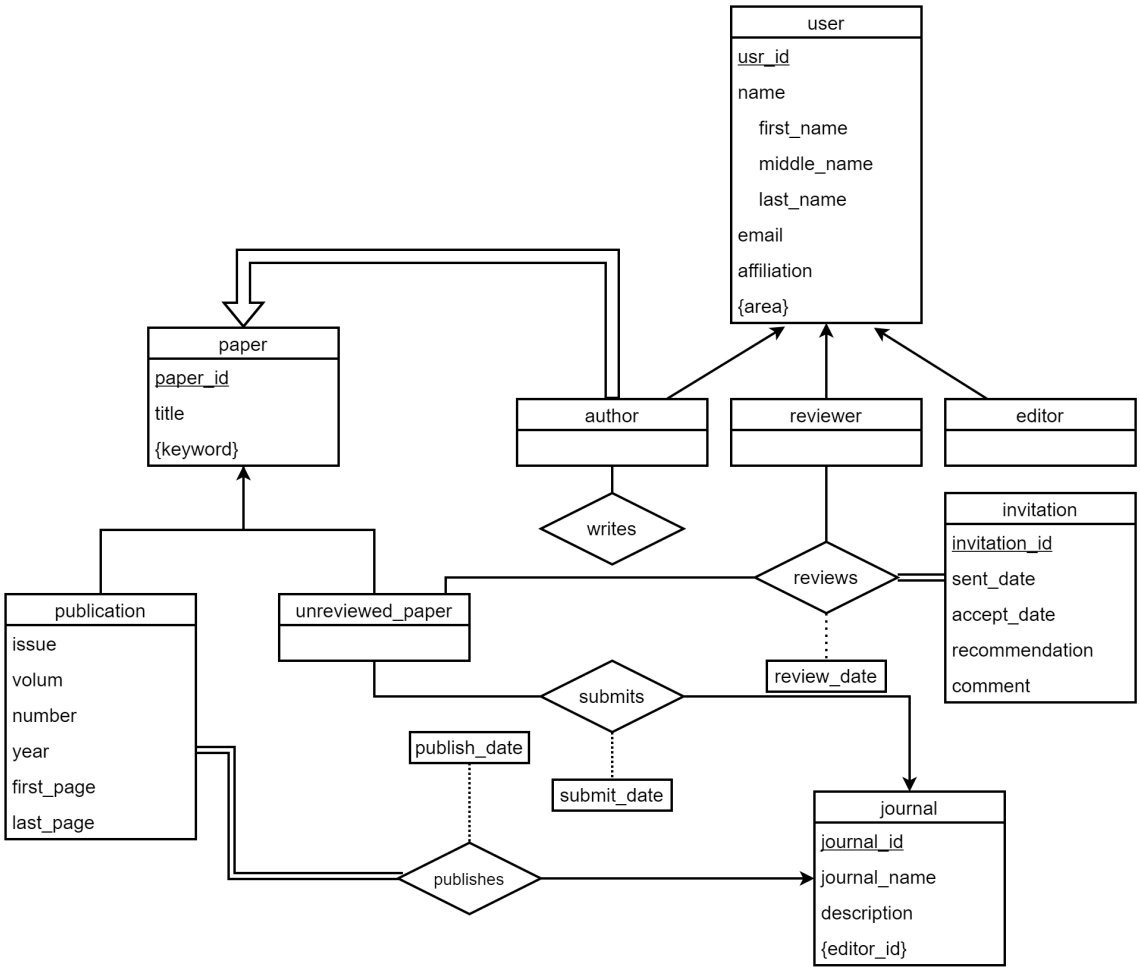
210016568

December 10, 2021

1 Database Modeling

1.1 Question a

The E-R diagram is shown below:



Assumptions I made:

- Paper will become publication after it is published.
- Publication is dependent on journal existence.
- Not all papers will be submitted.
- Once any paper is submitted, a reviewer will be invited to review the paper.
- This system has 3 log in interfaces for author, reviewer, editor.
- A reviewer or an editor may be good at many areas.

1.2 Question b

Rational schema:

Entity sets:
user(usr_id, first_name, middle_name, last_name, email, affiliation)

attribute	type	description	primary key	foreign key	not null
usr_id	CHAR(9)	a series of long number to identify entity	YES	NO	YES
first_name	VARCHAR(50)		NO	NO	YES
middle_name	VARCHAR(50)	maybe someone does not have middle name	NO	NO	NO
first_name	VARCHAR(50)		NO	NO	YES
email	VARCHAR(50)		NO	NO	YES
affiliation	VARCHAR(50)		NO	NO	YES

author(*author_id*)

attribute	type	description	primary key	foreign key	not null
author_id	CHAR(9)		YES	from user.usr_id	YES

reviewer(*reviewer_id*)

attribute	type	description	primary key	foreign key	not null
reviewer_id	CHAR(9)		YES	from user.usr_id	YES

editor(*editor_id*)

attribute	type	description	primary key	foreign key	not null
editor_id	CHAR(9)		YES	from user.usr_id	YES

areas(*usr_id*, area)

attribute	type	description	primary key	foreign key	not null
usr_id	CHAR(9)		YES	from user.usr_id	YES
area	VARCHAR(50)	The area of areas this one is good at	YES	NO	YES

paper(paper_id, title)

attribute	type	description	primary key	foreign key	not null
paper_id	CHAR(9)		YES	NO	YES
title	VARCHAR(80)		NO	NO	YES

keywords(*paper_id*, keyword)

attribute	type	description	primary key	foreign key	not null
paper_id	CHAR(9)		YES	from paper.paper_id	YES
keyword	VARCHAR(20)		YES	NO	YES

unreviewed_paper(*paper_id*)

attribute	type	description	primary key	foreign key	not null
paper_id	CHAR(9)		YES	from paper.paper_id	YES

publication(*paper_id*, issue, volum, number, year, first_page, last_page)

attribute	type	description	primary key	foreign key	not null
paper_id	CHAR(9)		YES	from paper.paper_id	YES
issue	INT		NO	NO	YES
volum	INT		NO	NO	YES
number	INT		NO	NO	YES
year	CHAR(4)		NO	NO	YES
first_page	INT		NO	NO	YES
last_page	INT		NO	NO	YES

journal(journal_id, journal_name, description)

attribute	type	description	primary key	foreign key	not null
journal_id	CHAR(9)		YES	NO	YES
journal_name	VARCHAR(80)		NO	NO	YES
description	VARCHAR(200)		NO	NO	YES

editors_of_journal(*journal_id*, *editor_id*)

attribute	type	description	primary key	foreign key	not null
journal_id	CHAR(9)		YES	from journal.journal_id	YES
editor_id	CHAR(9)		YES	from editor.editor_id	YES

invitation(invitation_id, send_date, accept_date, recommendation, comment)

attribute	type	description	primary key	foreign key	not null
invitation_id	CHAR(9)		YES	NO	YES
send_date	DATE		NO	NO	YES
accept_date	DATE		NO	NO	
recommendation	INT	0 and 1 represent reject and accept	NO	NO	
comment	VARCHAR(200)		NO	NO	YES

Relation:
writes(*author_id*, *paper_id*)

attribute	type	description	primary key	foreign key	not null
author_id	CHAR(9)		YES	from author.author_id	YES
paper_id	CHAR(9)		YES	from paper.paper_id	YES

submits(*paper_id*, *journal_id*, submit_date)

attribute	type	description	primary key	foreign key	not null
paper_id	CHAR(9)		YES	from unreviewed_paper.paper_id	YES
journal_id	CHAR(9)		YES	from journal.paper_id	YES
submit_date	DATE		NO	NO	YES

reviews(*reviewer_id*, *paper_id*, *invitation_id*, review_date)

attribute	type	description	primary key	foreign key	not null
reviewer_id	CHAR(9)		YES	from reviewer.reviewer_id	YES
paper_id	CHAR(9)		YES	from unreviewed_paper.paper_id	YES
invitation_id	CHAR(9)		YES	from invitation.invitation_id	YES
review_date	DATE		NO	NO	YES

publishes(*paper_id*, *journal_id*, publish_date)

attribute	type	description	primary key	foreign key	not null
paper_id	CHAR(9)		YES	from publication.paper_id	YES
journal_id	CHAR(9)		YES	from journal.journal_id	YES
publish_date	DATE		NO	NO	YES

1.3 Question c

(i)

```
SELECT
       name,
       description
FROM
       journal
ORDER BY name;
```

(ii)

```
SELECT
       d.name,
       d.affliation,
       concat(d.area) AS areas
From
       (journal
       NATURAL JOIN editor_of_journal
       LEFT JOIN user ON editor_of_journal.editor_id=user.usr_id
       LEFT JOIN areas ON editor_of_journal.editor_id=areas.usr_id) d
WHERE
       journal.name='Data Newsletter'
GROUP BY
       d.name;
```

(iii)

```
SELECT
        d.name,
        d.email,
        d.title
FROM
        (reviews
        NATURAL JOIN invitation
        NATURAL JOIN reviewer
        LEFT JOIN user ON reviewer.reviewer_id=user.usr_id
        LEFT JOIN unreviewed_paper ON reviews.paper_id=unreviewed_paper.paper_id) d
WHERE
        accept_date>'2019-12-31'
        AND
        accept_date<'2021-01-01'
        AND
        d.paper_id NOT IN
        (SELECT
                paper_id
        FROM
                publication);
```

2 SQL

before we write SQL code, we need to enforce foreign key constraints.

```
PRAGMA foreign_keys = TRUE;
```

2.1 Question a

```
CREATE TABLE person (
        person_id CHAR(9),
        name VARCHAR (50) NOT NULL,
        phone VARCHAR (10),
        email VARCHAR (50),
        PRIMARY KEY (person_id)
);

CREATE TABLE club (
        club_id CHAR (5),
        club_name VARCHAR(50) NOT NULL,
        leader_id CHAR(9),
        PRIMARY KEY (club_id),
        FOREIGN KEY (leader_id) REFERENCES person(person_id) ON DELETE SET NULL ON
            UPDATE CASCADE
);

CREATE TABLE club_member (
        club_id CHAR (5),
        person_id CHAR(9),
        join_date DATE,
        PRIMARY KEY (club_id,person_id),
        FOREIGN KEY (club_id) REFERENCES club(club_id) ON DELETE SET NULL ON UPDATE
            CASCADE,
        FOREIGN KEY (person_id) REFERENCES person (person_id) ON DELETE CASCADE ON
            UPDATE CASCADE
);

CREATE TABLE room (
        room_no VARCHAR (5),
        capacity INT NOT NULL,
        PRIMARY KEY (room_no)
);

CREATE TABLE club_time (
        club_id CHAR (5),
        day_of_week INT,
        room_no VARCHAR (5),
        time_start TIME,
        time_end TIME,
        PRIMARY KEY (club_id, day_of_week, room_no),
        FOREIGN KEY (club_id) REFERENCES club(club_id) ON DELETE SET NULL ON UPDATE
            CASCADE,
        FOREIGN KEY (room_no) REFERENCES room(room_no) ON DELETE CASCADE ON UPDATE
            CASCADE
);
```

2.2 Question b

I insert some data to let the output of queries can be seen:

```
INSERT INTO person VALUES
       ('200911001', 'A A', '7000000009', 'A@hotmail.com'),
       ('200911002', 'B B', '5434756346', 'B@hotmail.com'),
       ('210101001', 'C C', '6789765998', 'C@hotmail.com'),
       ('210203001', 'D D', '4758765675', 'D@hotmail.com'),
       ('210203002', 'E E', '7658768786', 'E@hotmail.com'),
       ('210308001', 'F F', '7346666544', 'F@hotmail.com'),
       ('210308002', 'G G', '9239873733', 'G@hotmail.com'),
       ('210308003', 'H H', '1759484444', 'H@hotmail.com'),
       ('210308004', 'I I', '9472223534', 'I@hotmail.com'),
       ('210308005', 'J J', '4892297522', 'J@hotmail.com');

INSERT INTO club VALUES
       ('19071', 'Club A', '200911001'),
       ('19072', 'Club B', '200911002');

INSERT INTO club_member VALUES
       ('19071', '210101001', '2021-01-01'),
       ('19072', '210203001', '2021-02-03'),
       ('19072', '210203002', '2021-02-03'),
       ('19071', '210308001', '2021-03-08'),
       ('19071', '210308002', '2021-03-08'),
       ('19072', '210308003', '2021-03-08'),
       ('19071', '210308004', '2021-03-08'),
       ('19072', '210308005', '2021-03-08');

INSERT INTO room VALUES
       ('1A', 1),
       ('2A', 2),
       ('3B', 4),
       ('4B', 1);

INSERT INTO club_time VALUES
       ('19071', 1, '1A', '9:00:00', '21:00:00'),
       ('19071', 2, '2A', '9:00:00', '21:00:00'),
       ('19072', 3, '3B', '9:00:00', '21:00:00'),
       ('19072', 4, '4B', '9:00:00', '21:00:00'),
       ('19072', 5, '3B', '9:00:00', '17:00:00');
```

(i) List names of clubs and their leaders, ordered by the club name

```
SELECT
       club_name,
       name
FROM
       club
       LEFT JOIN person
       ON club.leader_id=person.person_id
ORDER BY club_name;
```

(ii) List club names and the number of their members, listed in descending order by the number of members.

```
SELECT
        club_name,
        COUNT(person_id) AS number
FROM
        club
        NATURAL JOIN club_member
GROUP BY number;
```

(iii) List of email addresses of all people having some activity at room 3B on a Wednesday.

```
select DISTINCT
        email
FROM
        (room
        NATURAL JOIN club_time
        NATURAL JOIN club
    NATURAL JOIN club_member
        NATURAL JOIN person) d
WHERE
        room_no='3B'
        AND
        strftime('%w',d.join_date)
        IN
        (SELECT
                day_of_week
        FROM
                club_time
        WHERE
                room_no='3B'
        );
```

2.3 Question c

```
-- in SQLite, there is no FLOOR function,
-- I assume that we use SQL or SQL server
-- to run it, so that the code can run well
-- FLOOR(1.6)=1, FLOOR(1.2)=1, FLOOR(1)=1
UPDATE
        room
SET
        capacity = FLOOR(capacity/2)
WHERE
        room.capacity>=2;
```

2.4 Question d

Trigger is automatically run after setting it, it will happen when someone insert, update or delete.

When we insert new data into club, we need to create a triggle for checking whether the leader we will insert is the leader of other club's leader, if it is, we need to give a warning that this one cannot become the leader of this club.

When we insert a new data into club_time table, we need to create a trigger for checking if the day_of_week has already has a meeting, if it is, we need to give a warning that this day has plan a meeting and you cannot insert it.

3 Normalisation

3.1 Question a

(i)

Unnormalization: A table that may contain one or more repeating groups. My understanding is that the data is redundant.

1NF: A relation in which the intersection of each row and column contains one and only one value.

That means that for 1NF, only one data can appear at the intersection of the x-axis and y-axis of the table.

So, according to the definition, this data is unnormalized.

(ii)

The reason why it is not useful:

1. In relational databases, data tables that do not conform to 1NF cannot be created in RDBMS.
2. it is very convenient for humans to observe, but the computer cannot recognize the association.

(iii)

If we want the data become 1NF, we need to convert it as:

Acronym	ID	Funder	Budget	Staff name	Staff ID	FTE	Task	Task name	Month	Days
COD	62	EU	9500	Smith	2311	20%	T1	Lead	Dec-20	5
COD	62	EU	9500	Smith	2311	20%	T2	Code	Jan-21	5
COD	62	EU	9500	Lopez	5732	50%	T2	Code	Dec-20	7
COD	62	EU	9500	Lopez	5732	50%	T3	Docs	Dec-20	3
COD	62	EU	9500	Lopez	5732	50%	T2	Code	Jan-21	5
COD	62	EU	9500	Lopez	5732	50%	T2	Docs	Jan-21	5
BUG	41	UKRI	8000	Müller	4318	1.0	T1	Site	Jan-21	20
BUG	41	UKRI	8000	Lopez	5732	0.5	T2	DB	Jan-21	10
RAY	29	RSE	4000	Melnyk	3821	25%	T2	Tests	Jan-21	10

3.2 Question b

Functional Dependency: If A and B are attributes of relation R, B is functionally dependent on A (denoted $A \rightarrow B$), if each value of A in R is associated with exactly one value of B in R.

It is used to describe relationship between attributes.

It can be divided into **Full Functional Dependency** and **Transitive Dependency**

Fully Functional Dependency means let A and B are attributes of a relation. B is fully functionally dependent on A, if B is functionally dependent on A, but not on any proper subset of A.

Transitive Dependency means let A, B, and C are attributes of a relation. If $A \rightarrow B$ and $B \rightarrow C$ then C is said to be transitively dependent on A via B.

Combined with the understanding of the scenario, the ID of project functionally determines the Acronym, funder and budget of project.

every staff has a unique Staff ID, so Staff ID functionally determines the name of staff and FTE.

For the rest part, different staff has different task to do, although the same task may have different task name, although task name and month seem like can determine the days, but it is not in line with common sense, so I regard it as a coincidence.

Finally, the minimum functional dependency set I identified is shown below:

$$ID \rightarrow Acronym, Funder, Budget$$

$$Staff\ ID \rightarrow Staff\ name, FTE$$

$$ID, Staff\ ID, Task, Task\ name, Month \rightarrow Days$$

The primary key is $(ID, StaffID, Task, Taskname, Month)$

3.3 Question c

Yes, there are some update anomalies.

For example, if I want to delete the project 'RAY', I will lose the data of the staff named Melnyk.

3.4 Question d

2NF means A relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on any candidate key.

So after make sure the every non-primary-key attribute is fully functionally dependent on any candidate key, the 2NF should be that:

$(\underline{ID}, Acronym, Funder, Budget)$

$(\underline{Staff\ ID}, Staff\ name, FTE)$

$(\underline{ID, Staff\ ID, Task}, Taskname, \underline{Month}, Days)$

3NF means A relation that is in first and second normal form and in which no non-primary-key attribute is transitively dependent on any candidate key.

After checking, the table has meet the demand of 3NF.

3.5 Question e

if it uses the general definition of 3NF, because the Acronym, ID FUnder, BUdget can determine each other, so we need to select an attribute (I choosed ID) to determine others.

i.e.,

$(\underline{ID}, Acronym)$

$(\underline{ID}, Funder)$

$(\underline{ID}, Budget)$