DATABASE

Team Project – Summer 2022

Project title: Journal of IoT Research

Teacher: Nagat Drawel

Teammates:

Chester Rae De Vera

Heping Song

Snehalata Murmu

Yateng Geng

Date:2022-10-09

CATALOG

Note: based on the feedback from Deliverable 2, we only updated the comments for the indexes

INTRODUCTION

In this project, our group developed a miniature database system, and evaluate several queries and transactions against the database.

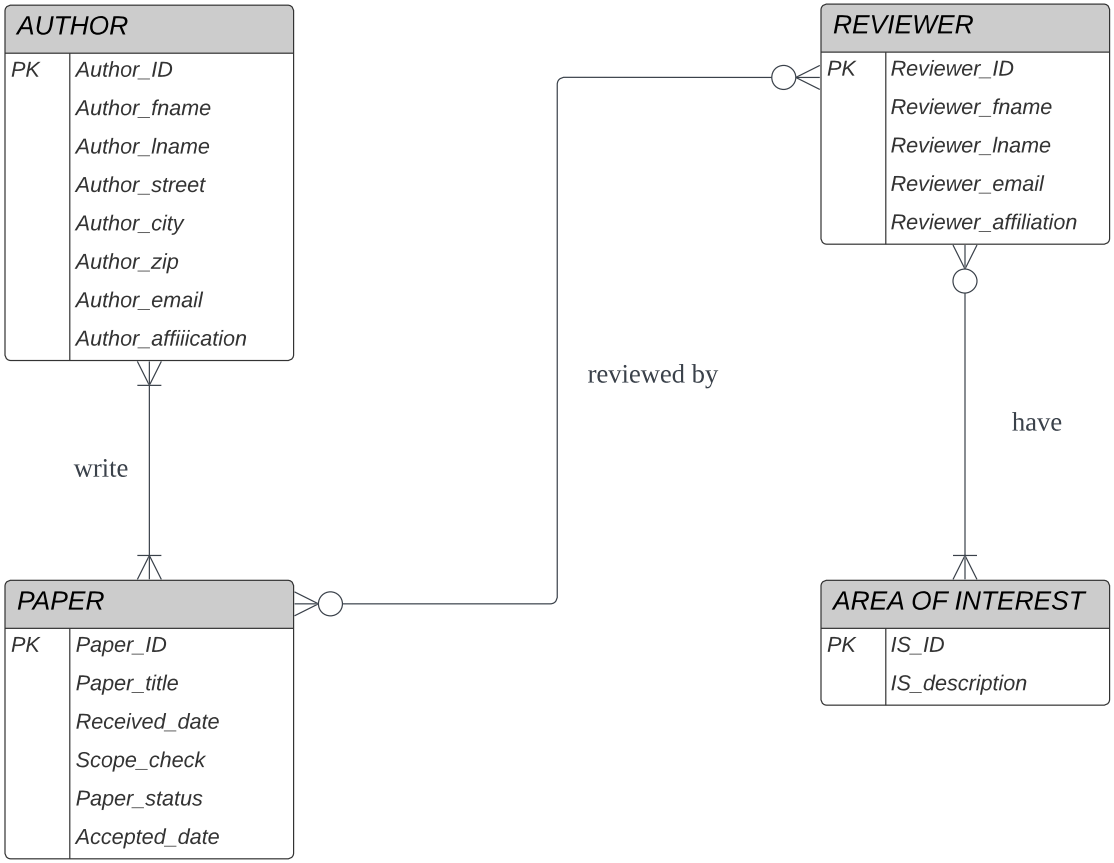
Deliverable 1 is basically about conceptual database design, entity relations, 3NF, etc.

Deliverable 2 is mostly about building the database on SQL server and insert data into it. Besides, we are using sequence, index, and other DML or DDL to make it from concept to practical database system.

Deliverable 3 is about using premium SQL techniques to achieve business use, such as views, stored procedures, and triggers.

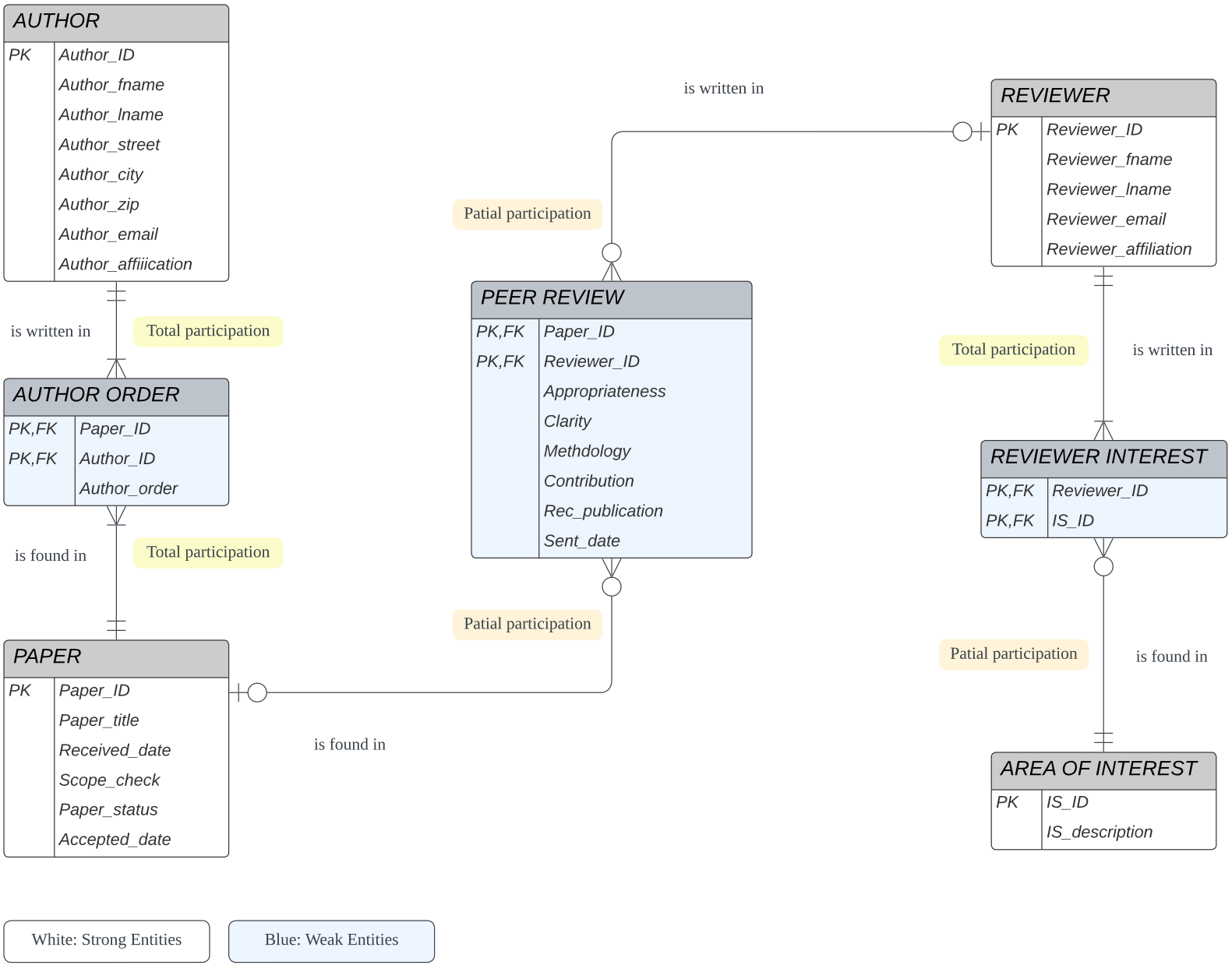
Conceptual DB Design

Chester Rae De Vera，Heping Song，Snehalata Murmu，Yateng Geng

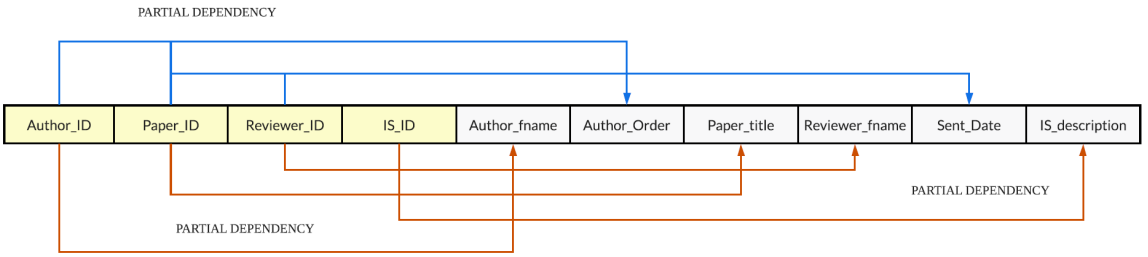


Group Assignment: JIR System

Chester Rae De Vera, Heping Song, Snehalata Murmu, Yateng Geng



Dependency Diagram (to 2nd form)



New Dependency Diagrams (2nd form)

Author_ID	Author_fname
Paper_ID	Paper_title
Reviewer_ID	Reviewer_fname
IS_ID	IS_description

Author_ID	Paper_ID	Author_Order
Paper_ID	Reviewer_ID	Sent_Date
Reviewer_ID	IS_ID	

Final Dependency Diagrams (3rd form)

Author_ID	Author_fname	.....	**AUTHOR**
Paper_ID	Paper_title	.....	**PAPER**
Reviewer_ID	Reviewer_fname	.....	**REVIEWER**
Author_ID	Paper_ID	Author_Order	**AUTHOR ORDER**
Paper_ID	Reviewer_ID	Sent_Date	**PEER REVIEW**
Reviewer_ID	IS_ID		**REVIEWER INTEREST**
IS_ID	IS_description		**AREA OF INTEREST**

```
--Team Project


--*Deliverable #3: Query Development starts from Ln206


/*1. Create a database schema for your project using SQL DDL (Data Definition Language) statements.

Choose appropriate data types for each attribute and include Primary Key and Foreign Key constraints, Check, Unique,

and Not Null constraints. Please assume defaults for ON DELETE clauses for FOREIGN KEYs.*/

CREATE TABLE AUTHOR

(AUTHOR_ID INT CONSTRAINT AUTH_ID_PK PRIMARY KEY,

AUTHOR_FNAME VARCHAR(30) NOT NULL,

AUTHOR_LNAME VARCHAR(30) NOT NULL,

AUTHOR_STREET VARCHAR(50),

AUTHOR_CITY VARCHAR(50),

AUTHOR_ZIP CHAR(7),

AUTHOR_EMAIL VARCHAR(30) NOT NULL UNIQUE,

AUTHOR_AFFILIATION VARCHAR(50));


CREATE TABLE REVIEWER

(REVIEWER_ID INT CONSTRAINT REV_ID_PK PRIMARY KEY,

REVIEWER_FNAME VARCHAR(30) NOT NULL,

REVIEWER_LNAME VARCHAR(30) NOT NULL,

REVIEWER_EMAIL VARCHAR(30) NOT NULL UNIQUE,

REVIEWER_AFFILIATION VARCHAR(50));


CREATE TABLE AREA_OF_INTEREST

(IS_ID VARCHAR(10),

IS_DESCRIPTION VARCHAR(30) NOT NULL,

CONSTRAINT IS_ID_PK PRIMARY KEY(IS_ID));
```

```
CREATE TABLE PAPER

(PAPER_ID INT CONSTRAINT PAPER_ID_PK PRIMARY KEY,

PAPER_TITLE VARCHAR(40) NOT NULL,

RECEIVED_DATE DATE,

SCOPE_CHECK VARCHAR(10) NOT NULL,

PAPER_STATUS VARCHAR(12) NOT NULL,

ACCEPTED_DATE DATE,

CHECK (PAPER_STATUS IN ('received','rejected','under review', 'accepted')));


CREATE TABLE AUTHOR_ORDER

(PAPER_ID INT CONSTRAINT AU_ORDER_PAPER_ID_FK FOREIGN KEY(PAPER_ID) REFERENCES PAPER ON
DELETE CASCADE,

AUTHOR_ID INT CONSTRAINT AU_ORDER_AU_ID_FK FOREIGN KEY(AUTHOR_ID) REFERENCES AUTHOR
ON DELETE CASCADE,

AUTHOR_ORDER NUMERIC(2),

PRIMARY KEY (PAPER_ID, AUTHOR_ID));


CREATE TABLE PEER_REVIEW

(PAPER_ID INT CONSTRAINT PEER_REV_PAPER_ID_FK FOREIGN KEY(PAPER_ID) REFERENCES PAPER ON
DELETE CASCADE,

REVIEWER_ID INT CONSTRAINT PEER_REV_REV_ID_FK FOREIGN KEY(REVIEWER_ID) REFERENCES
REVIEWER ON DELETE CASCADE,

APPROPRIATENESS NUMERIC(2),

CLARITY NUMERIC(2),

METHODOLOGY NUMERIC(2),

CONTRIBUTION NUMERIC(2),

REC_PUBLICATION VARCHAR(7),

SENT_DATE DATE

PRIMARY KEY (PAPER_ID, REVIEWER_ID));
```

```
CREATE TABLE REVIEWER_INTEREST

(REVIEWER_ID INT CONSTRAINT REV_INT_PAPER_ID_FK FOREIGN KEY(REVIEWER_ID) REFERENCES
REVIEWER ON DELETE CASCADE,

IS_ID VARCHAR(10) CONSTRAINT REV_INT_IS_ID_FK FOREIGN KEY(IS_ID) REFERENCES
AREA_OF_INTEREST ON DELETE CASCADE,

PRIMARY KEY (REVIEWER_ID,IS_ID));

/*2. Explain why you chose your ON DELETE actions. As an example: “if we delete a customer then we
delete their accounts”).*/

--WE CHOSE ON DELETE CASCADE BECAUSE IF WE WANT TO DELETE THE
AUTHOR/PAPER/REVIEWER/AREA_OF_INTEREST

--THEN WE DELETE EVERYTHING INCLUDING THE FOREIGN KEYS FROM
AUTHOR_ORDER/PEER_REVIEW/REVIEWER_INTEREST.



/*3. Populate every relation with sufficient representative rows using DML (Data Manipulation
Language) statements.*/

--Simulation data

INSERT INTO AUTHOR

(AUTHOR_ID,AUTHOR_FNAME,AUTHOR_LNAME,AUTHOR_STREET,AUTHOR_CITY,AUTHOR_ZIP,AUTHOR
_EMAIL,AUTHOR_AFFILIATION)

VALUES

(1000,'JANE','AUSTIN','BALL ST.','MONTREAL','H8N1X1','PRIDE77@GMAIL.COM','Montreal University'),

(1001,'JAKE','SPARROW','PARK ST.','ONTARIO','H2J7Y2','BLACKPEARL@HOTMAIL.COM','Disney'),

(1002,'PHOEBE','BUFFAY','VICTORIA ST','QUEBEC','H3D6T8','SMELLYCAT1@GMAIL.COM','Central Park'),

(1003,'JOHN','SNOW','ROCKWELL ST.','TORONTO','H8A4V3','I_KNOW_THINGS@GMAIL.COM','The Wall'),

(1004,'TIM','COOK','VALLEY ST.','KANSAS','H4G3Z4','IPHONE184673@GMAIL.COM','APPLE');


INSERT INTO REVIEWER

(REVIEWER_ID,REVIEWER_FNAME,REVIEWER_LNAME,REVIEWER_EMAIL,REVIEWER_AFFILIATION)

VALUES
```

(284,'BOJACK','HORSEMAN','JNGJAHK@GMAIL.COM','Barley Hill Primary School'),

(285,'PETER','PARKER','PETTT@HOTMAIL.COM','Marvel'),

(286,'TIM','HORTONS','TIMMMH@HOTMAIL.COM','Mcgill University'),

(287,'KFC','KENTUCKY','KFC111@GMAIL.COM','Concordia University'),

(288,'AVRIL','LAVIGNE','SKATORBOY7@GMAIL.COM','University of Toronto');

INSERT INTO AREA_OF_INTEREST

(IS_ID,IS_DESCRIPTION)

VALUES

('IS2003','database modeling'),

('IS2004','artificial intelligence'),

('IS2005','engineering mathematics'),

('IS2006','speech enhancement'),

('IS2007','data security');

INSERT INTO PAPER

(PAPER_ID,PAPER_TITLE,RECEIVED_DATE,SCOPE_CHECK,PAPER_STATUS,ACCEPTED_DATE)

VALUES

(48473,'The Importance of Family Ties','2019-04-28','pass','received','2019-04-29'),

(48474,'Prostitution Should Never be Legalized','2019-04-29','fail','received','2019-05-14'),

(48475,'It is just a Painting: When Art Matters','2019-05-04','fail','Received','2019-06-01'),

(48476,'The Drinking Age should be Higher','2019-06-01','fail','RECEIVED','2019-06-21'),

(48477,'The Trojan Horse of Data Secuirty','2019-07-28','pass','received','2019-08-28');

INSERT INTO AUTHOR_ORDER

(PAPER_ID,AUTHOR_ID,AUTHOR_ORDER)

VALUES

(48473,1003,1),

(48473,1000,2),

```
(48475,1002,1),

(48475,1004,2),

(48477,1001,1),

(48477,1004,2),

(48477,1003,3);


INSERT INTO PEER_REVIEW

(PAPER_ID,REVIEWER_ID,APPROPRIATENESS,CLARITY,METHODOLOGY,CONTRIBUTION,REC_PUBLICATIO
N,SENT_DATE)

VALUES

(48473,285,7,8,8,5,'ACCEPT','2019-05-03'),

(48473,286,6,7,7,7,'ACCEPT','2019-05-04'),

(48473,287,7,8,8,5,'ACCEPT','2019-05-06'),

(48477,284,9,6,8,8,'ACCEPT','2019-05-02'),

(48477,285,8,8,8,8,'ACCEPT','2019-05-03'),

(48477,288,7,7,8,8,'ACCEPT','2019-05-05');


INSERT INTO REVIEWER_INTEREST

(REVIEWER_ID,IS_ID)

VALUES

(284,'IS2004'),

(284,'IS2007'),

(285,'IS2003'),

(285,'IS2005'),

(286,'IS2004'),

(286,'IS2006');


SELECT * FROM PAPER;

SELECT * FROM AUTHOR;
```

```
SELECT * FROM REVIEWER;

SELECT * FROM AREA_OF_INTEREST;

SELECT * FROM AUTHOR_ORDER;

SELECT * FROM PEER_REVIEW;

SELECT * FROM REVIEWER_INTEREST;



/*4.Create at least one SEQUENCE to generate values for one of the tables of your choice.

Select the start value of your sequence and the increment by value.

Do not cache any values, also do not cycle.

You can add more sequences if you like.*/

CREATE SEQUENCE AUTHOR_NUM_SEQ AS INT

START WITH 1005

INCREMENT BY 1

MINVALUE 1005

MAXVALUE 999999

NO CYCLE

NO CACHE;


--Check created SEQUENCE

SELECT * FROM sys.sequences WHERE name = 'AUTHOR_NUM_SEQ';

--Test created SEQUENCE

INSERT INTO AUTHOR

VALUES (NEXT VALUE FOR
AUTHOR_NUM_SEQ,'Walter','White',NULL,NULL,'H5J1Y7','couldbeworse@gmail.com','J. P. Wynne High
School');

SELECT * FROM AUTHOR;


/*5. Create two indexes to your database. Select the one that you think will be beneficial!
```

It is not critical that these be the most important indexes,

but the choices should make good sense in terms of the queries (and inserts, deletes, and updates)

that you expect will be run on the database*/

--INDEX

--UPDATE：

--I choice those Emails as indexes because they are UNIQUE and NOT NULL;

--And normally emails are related to account management, so it's possible to use they for editor search work

CREATE INDEX INDEX_AUTHOR_EMAIL

ON AUTHOR (AUTHOR_EMAIL);

CREATE INDEX INDEX_REVIEWER_EMAIL

ON REVIEWER (REVIEWER_EMAIL);

/*6. Write at least two ALTER statements to either add/modify columns or add constraints to any of the tables.*/

--Add columns

ALTER TABLE REVIEWER

ADD MOBILE VARCHAR(15);

SELECT * FROM REVIEWER;

--Modify columns and add constraint

ALTER TABLE PAPER

ALTER COLUMN RECEIVED_DATE DATE NOT NULL;

ALTER TABLE PEER_REVIEW

ADD CONSTRAINT CHECK_APPROPRIATENESS CHECK(APPROPRIATENESS BETWEEN 1 AND 10)

```
ALTER TABLE PEER_REVIEW
ADD CONSTRAINT CHECK_CLARITY CHECK(CLARITY BETWEEN 1 AND 10)

ALTER TABLE PEER_REVIEW
ADD CONSTRAINT CHECK_METHODOLOGY CHECK(METHODOLOGY BETWEEN 1 AND 10)

ALTER TABLE PEER_REVIEW
ADD CONSTRAINT CHECK_CONTRIBUTION CHECK(CONTRIBUTION BETWEEN 1 AND 10)

--Deliverable #3: Query Development

/*1. Develop at least 3 complex queries and 2 views.
SELECT queries that represent answers to likely business questions to be faced by the users of your database system.*/

--Qurry 1
--Editor need to choose the papers that are out of slope or rejected and send email to the authors
SELECT O.PAPER_ID,
P.PAPER_TITLE,P.RECEIVED_DATE,A.AUTHOR_ID,A.AUTHOR_FNAME,A.AUTHOR_LNAME,A.AUTHOR_EMAIL
FROM PAPER P JOIN AUTHOR_ORDER O
ON P.PAPER_ID = O.PAPER_ID
JOIN AUTHOR A
ON O.AUTHOR_ID = A.AUTHOR_ID
WHERE SCOPE_CHECK = 'fail' or PAPER_STATUS = 'rejected'
ORDER BY RECEIVED_DATE;

--Qurry 2
--Editor need to know the papers that need to be distributed to reviewers
```

```
SELECT PAPER_ID,PAPER_TITLE,RECEIVED_DATE,PAPER_STATUS

FROM PAPER

WHERE PAPER_ID NOT IN (SELECT PAPER_ID

FROM PEER_REVIEW)

AND PAPER_STATUS != 'rejected';


--Qurry 3

--Editor need to find the new reviewers who don't have an area of interest yet, and give them at least one IS_ID

SELECT REVIEWER_ID,REVIEWER_FNAME,REVIEWER_LNAME,REVIEWER_AFFILIATION

FROM REVIEWER

WHERE REVIEWER_ID NOT IN (SELECT REVIEWER_ID

FROM REVIEWER_INTEREST);


--View 1

--Work summary from the one of the reviewer's perspective;

--Reviewer could have an idea about the work been done and the word need to by done, and his/her grading style in general;

--Editor could also use this view to check review's work, and check if there are something wrong, like whether the pass_rate is too high

GO

CREATE VIEW REVIEWER_GENERAL_285

AS

 SELECT

 REVIEWER_ID,

 SUM(CASE WHEN REC_PUBLICATION IS NOT NULL THEN 1 ELSE 0 END) AS PAPER_FINISHED,

 SUM(CASE WHEN REC_PUBLICATION = 'ACCEPT' THEN 1 ELSE 0 END) AS PAPER_ACCEPTED,

 FORMAT(SUM(CASE WHEN REC_PUBLICATION = 'ACCEPT' THEN 1 ELSE 0 END)/SUM(CASE WHEN REC_PUBLICATION IS NOT NULL THEN 1 ELSE 0 END),'P')

 AS PASS_RATE,
```

```
AVG(APPROPRIATENESS+CLARITY+METHODOLOGY+CONTRIBUTION) AS AVG_POINTS,
SUM(CASE WHEN REC_PUBLICATION IS NULL THEN 1 ELSE 0 END) AS PAPER_ONGO
FROM PEER_REVIEW
WHERE REVIEWER_ID = '285'
GROUP BY REVIEWER_ID;
GO
--For testing
SELECT * FROM REVIEWER_GENERAL_285;


--View 2
--Word need to be done from the one of the reviewer's perspective, he/she should have an direct view for it;
--Yet he/she shouldn't have access to the Authors' information, for security reason and it's irrelevant for judging, that's why we use VIEW;
--We could even build more bussness rules around this view, i.e. papers must be reviewed in 30 days after it's sent to reviewers


--Adding more data for testing
INSERT INTO PEER_REVIEW
(PAPER_ID,REVIEWER_ID,APPROPRIATENESS,CLARITY,METHODOLOGY,CONTRIBUTION,REC_PUBLICATION,SENT_DATE)
VALUES
(48475,286,NULL,NULL,NULL,NULL,NULL,'2019-05-07'),
(48476,286,6,7,NULL,NULL,NULL,'2019-06-03');
SELECT * FROM PAPER;
SELECT * FROM PEER_REVIEW;


GO
CREATE VIEW REVIEWER_ONGOING_286
AS
```

```
SELECT
P.PAPER_ID,P.PAPER_TITLE,P.RECEIVED_DATE,APPROPRIATENESS,CLARITY,METHODOLOGY,CONTRIBUTI
ON,REC_PUBLICATION,
SENT_DATE, DATEDIFF(DAY,SENT_DATE,GETDATE()) AS WAITING_DAYS
FROM PAPER P JOIN PEER_REVIEW R
ON P.PAPER_ID = R.PAPER_ID
WHERE APPROPRIATENESS IS NULL OR CLARITY IS NULL OR METHODOLOGY IS NULL OR
METHODOLOGY IS NULL OR CONTRIBUTION IS NULL OR REC_PUBLICATION IS NULL
AND REVIEWER_ID = 286;
GO
--For testing
SELECT * FROM REVIEWER_ONGOING_286;
```

/*Create 2 stored procedures that enact business rules that must be supported by the database
(for example, to allow the user to insert, delete or update through the stored procedures).
At lease one of the stored procedures must use parameters and use conditional logic.
Explain the purpose of each of the stored procedures.*/

```
--Procedure 1
--After sending the papers to reviews, we need to INSERT new record in the table of PEER_REVIEW;
--and the PAPER_STATUS should be UPDATED to under_review;
GO
CREATE PROCEDURE PRC_SENDING_PAPERS @P_ID INT, @R_ID INT
AS
BEGIN
  INSERT INTO PEER_REVIEW (PAPER_ID,REVIEWER_ID,SENT_DATE)
```

```
  VALUES
  (@P_ID,@R_ID,GETDATE());
  PRINT 'Paper ' + CAST(@P_ID AS CHAR) + 'sent to reviewer '+ CAST(@R_ID AS CHAR)
  UPDATE PAPER
  SET PAPER_STATUS = 'under review'
  WHERE PAPER_ID = @P_ID;
END;
GO

--For testing
EXEC PRC_SENDING_PAPERS 48475,285;
SELECT * FROM PEER_REVIEW;
SELECT * FROM PAPER;

--Procedure 2
--When the paper is done reviewing, we need to decide wether to publish this paper or not;
--Assume the business rule here is: only the papers with avg total score above 30 and ACCEPTED by at least 3 reviewers could be published;
GO
CREATE PROCEDURE PRC_PAPER_FINALPASS @PA_ID INT
AS
BEGIN
   IF (SELECT AVG(APPROPRIATENESS+CLARITY+METHODOLOGY+CONTRIBUTION) FROM PEER_REVIEW WHERE PAPER_ID = @PA_ID) >= 30
          AND (SELECT SUM(CASE WHEN REC_PUBLICATION = 'ACCEPT' THEN 1 ELSE 0 END) FROM PEER_REVIEW WHERE PAPER_ID = @PA_ID) >=3

            SELECT PAPER_ID,AVG(APPROPRIATENESS+CLARITY+METHODOLOGY+CONTRIBUTION) AS AVG_SCORE,
            SUM(CASE WHEN REC_PUBLICATION = 'ACCEPT' THEN 1 ELSE 0 END) AS ACCEPT_NUM
```

```
        FROM PEER_REVIEW
        WHERE PAPER_ID = @PA_ID
        GROUP BY PAPER_ID;

    ELSE
    PRINT 'This paper should not be accepted or is still under review.';
END;
GO

--For testing
--The paper with avg total score above 30 and ACCEPTED by at least 3 reviewers
EXEC PRC_PAPER_FINALPASS 48477;
--The paper with avg total score under 30;
EXEC PRC_PAPER_FINALPASS 48473;
--The paper ACCEPTED by less than 3 reviewers;
EXEC PRC_PAPER_FINALPASS 48475;
```

/*3. Write a database trigger that change the status of the research paper based on the reviewers' feedbacks.

Your trigger must satisfy the following requirements:

a) Before making an overall decision, the number of reviewers for each research paper must be 2 or more, otherwise,

a message has to be displayed for the editor to invite an additional reviewer to get an extra opinion before making a decision.

b) Get the feedback rating points from each reviewer on the paper. If the average scale for appropriateness,

clarity, methodology, and contribution to the field that is given to the paper is more than 5-points,

then the status of the paper has to be changed to "accepted" and the date of acceptance is recorded.

```
If the scale less than 5-points the status is changed to “rejected.”
c) Test your trigger in all cases*/
GO
CREATE TRIGGER TRG_FEEDBACK_STATUS_SWITCH
ON PEER_REVIEW
AFTER INSERT,UPDATE
AS
DECLARE @PAP_ID INT
DECLARE @COUNT INT
DECLARE @AVERAGE NUMERIC(5,2)
BEGIN
SELECT @PAP_ID = PAPER_ID FROM inserted
SELECT @COUNT = COUNT(*) FROM PEER_REVIEW
WHERE PAPER_ID = @PAP_ID
SELECT @AVERAGE = AVG(APPROPRIATENESS+CLARITY+METHODOLOGY+CONTRIBUTION)
FROM PEER_REVIEW WHERE PAPER_ID = @PAP_ID
IF @COUNT < 2
PRINT 'Please invite an additional reviewer to get an extra opinion before making a decision.'
ELSE IF ((@COUNT >= 2) AND (@AVERAGE/4 >= 5))
UPDATE PAPER
SET PAPER_STATUS = 'ACCEPTED', ACCEPTED_DATE = GETDATE()
WHERE PAPER_ID = @PAP_ID
ELSE
UPDATE PAPER
SET PAPER_STATUS = 'REJECTED'
WHERE PAPER_ID = @PAP_ID
END;
GO
```

```
--For testing: this trigger is not done correctly, need to fix it
/*a) If the reviewer num is still less than 2*/
INSERT INTO PEER_REVIEW VALUES(48474,284,5,5,5,5,NULL,GETDATE());

/*b) If the reviewer num is 2 or more but the avg_score is not above 5*/
SELECT * FROM PAPER
SELECT * FROM PEER_REVIEW
INSERT INTO PEER_REVIEW VALUES(48474,287,3,3,3,3,NULL,GETDATE());
SELECT * FROM PAPER
SELECT * FROM PEER_REVIEW

/*C) If the reviewer num is 2 or more and the avg_score is above 5*/
SELECT * FROM PAPER
SELECT * FROM PEER_REVIEW
INSERT INTO PEER_REVIEW VALUES(48477,287,6,6,6,6,'ACCEPT',GETDATE());
SELECT * FROM PAPER
SELECT * FROM PEER_REVIEW
```