CS669

SiCheng Yi

LAB4

**Section One – Stored Procedures**

1. Create the tables in the social networking schema, including all of their columns, datatypes, and constraints. Populate the tables with data, ensuring that there are at least 5 people, at least 8 posts, and at least 4 likes. Most of the fields are self-explanatory. As far as the “content” field in Post, make them whatever you like, such as “Take a look at these new pics” or “Just arrived in the Bahamas”, and set the summary as the first 12 characters of the content, followed by “…”.

Create Table person(

person\_id decimal(12) PRIMARY KEY NOT NULL,

first\_name varchar(32) NOT NULL,

last\_name varchar(32) NOT NULL,

username varchar(20) NOT NULL);

Create Table post(

post\_id decimal(12) PRIMARY KEY NOT NULL,

person\_id decimal(12) NOT NULL,

content varchar(225) NOT NULL,

created\_on DATE NOT NULL,

summary varchar(15) NOT NULL,

FOREIGN KEY(person\_id) REFERENCES person(person\_id));

Create Table likes(

likes\_id decimal(12) PRIMARY KEY NOT NULL,

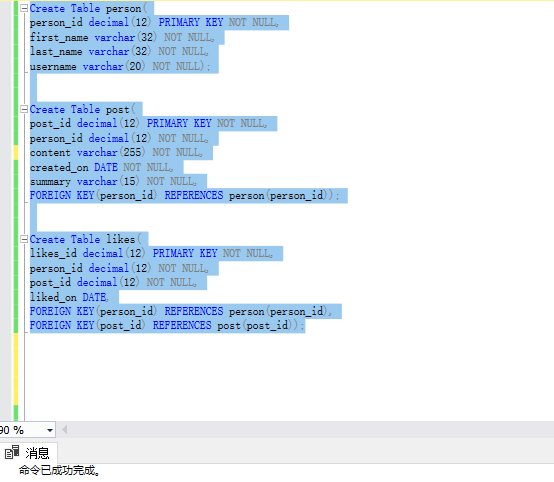
person\_id decimal(12) NOT NULL,

post\_id decimal(12) NOT NULL,

liked\_on DATE,

FOREIGN KEY(person\_id) REFERENCES person(person\_id),

FOREIGN KEY(post\_id) REFERENCES post(post\_id));



--person

Insert into person Values(1,'SiCheng','Yi','masteryi');

Insert into person Values(2,'Brian','Todd','todd');

Insert into person Values(3,'Judy','Date','judy');

Insert into person Values(4,'LiPing','He','pingping');

Insert into person Values(5,'Peter','Marsh','peter');

--post

Insert into post Values(1,1,'I Like This', CAST('4/1/2015' AS DATE),'There is a t...');

Insert into post Values(2,1,'Do not do this', CAST('8/3/2018' AS DATE),'These people...');

Insert into post Values(3,2,'Look at these animals', CAST('9/7/2017' AS DATE),'I like this ...');

Insert into post Values(4,3,'Big News', CAST('3/8/2020' AS DATE),'Boston has l...');

Insert into post Values(5,3,'Enjoy it', CAST('2/7/2012' AS DATE),'Tell us the ...');

Insert into post Values(6,4,'I like playing soccer', CAST('1/8/2020' AS DATE),'I like playi...');

Insert into post Values(7,4,'New semester is comming', CAST('9/6/2021' AS DATE),'Hello everyo...');

Insert into post Values(8,5,'It snows', CAST('12/7/2018' AS DATE),'Good afterno...');

--likes

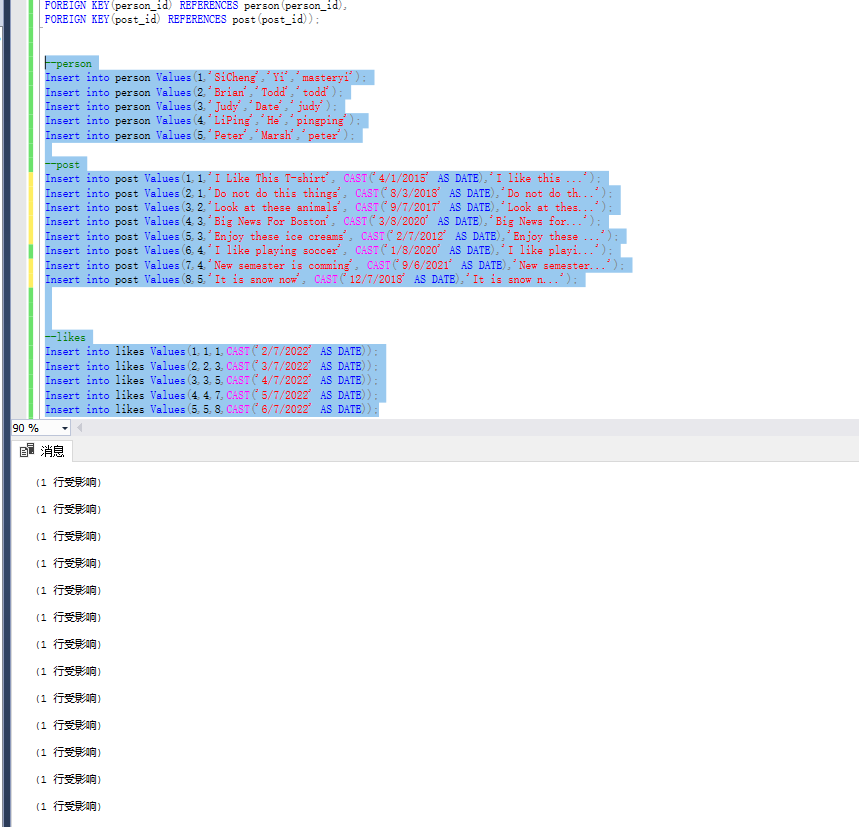
Insert into likes Values(1,1,1,CAST('2/7/2022' AS DATE));

Insert into likes Values(2,2,3,CAST('3/7/2022' AS DATE));

Insert into likes Values(3,3,5,CAST('4/7/2022' AS DATE));

Insert into likes Values(4,4,7,CAST('5/7/2022' AS DATE));

Insert into likes Values(5,5,8,CAST('6/7/2022' AS DATE));



1. Create a stored procedure named “add\_zana\_sage” which adds a person named “Zana Sage” to the Person table, then execute the stored procedure. List out the rows in the Person table to show that Zana Sage has been added.

Create Procedure add\_zana\_sage

AS

BEGIN

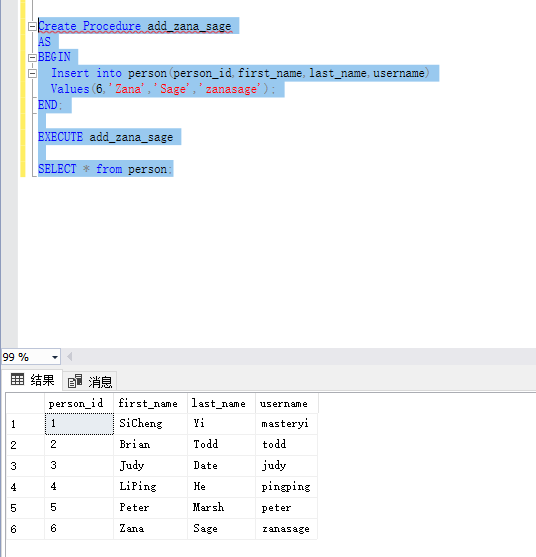
Insert into person(person\_id,first\_name,last\_name,username)

Values(6,'Zana','Sage','zanasage');

END;

EXECUTE add\_zana\_sage

SELECT \* from person;



1. Attempt to execute the “add\_zana\_sage” procedure a second time. Summarize what the issue is from the error that occurs as a result.



The error told us it tries to make another same primary key to make the line, but the primary key is unique, so the primary key constraint makes it error.

1. Create a reusable stored procedure named “add\_person” that uses parameters and allows you to insert any new person into the Person table. Execute the stored procedure with a person of your choosing, then list out the Person table to show that the person was added to the table.

Create Procedure add\_person

@usid DECIMAL,

@FNAME VARCHAR(30),

@LNAME VARCHAR(40),

@UNAME VARCHAR(40)

AS

BEGIN

Insert into person(person\_id,first\_name,last\_name,username)

VALUES(@usid,@FNAME,@LNAME,@UNAME);

END;

EXECUTE add\_person 7,'Mary','Smith','marysmith';

SELECT \* from person;



1. Create a reusable stored procedure named “add\_post” that uses parameters and allows you to insert any new post into the Post table. Instead of passing in the summary as a parameter, derive the summary from the content, storing the derivation temporarily in a variable (which is then used as part of the insert statement). Recall that the summary field stores the first 12 characters of the content followed by “…”. Execute the stored procedure to add a post of your choosing, then list out the Post table to show that the addition succeeded.

Create Procedure add\_post

@poid DECIMAL(12),

@persid DECIMAL(12),

@content VARCHAR(64),

@creatd DATE

AS

BEGIN

DECLARE @sum VARCHAR(20)

SET @sum = CONCAT(SUBSTRING(@content,1,12),'...')

INSERT INTO post(post\_id,person\_id,content,created\_on,summary)

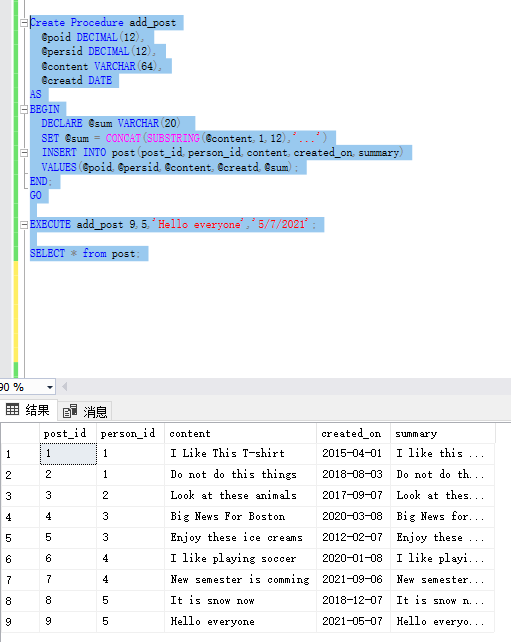
VALUES(@poid,@persid,@content,@creatd,@sum);

END;

GO

EXECUTE add\_post 9,5,'Hello everyone','5/7/2021';

SELECT \* from post;



1. Create a reusable stored procedure named “add\_like” that uses parameters and allows you to insert any new “like”. Rather than passing in the person\_id value as parameters to identify which person is liking which post, pass in the username of the person. The stored procedure should then lookup the person\_id and store it in a variable to be used in the insert statement. Execute the procedure to add a “like” of your choosing, then list out the Like table to show the addition succeeded.

Create Procedure add\_like

@liid DECIMAL(12),

@poid DECIMAL(12),

@username VARCHAR(20),

@lion DATE

AS

BEGIN

DECLARE @v\_personid DECIMAL(12);

SELECT @v\_personid = person\_id

FROM person

WHERE username = @username;

Insert into likes(likes\_id,person\_id,post\_id,liked\_on)

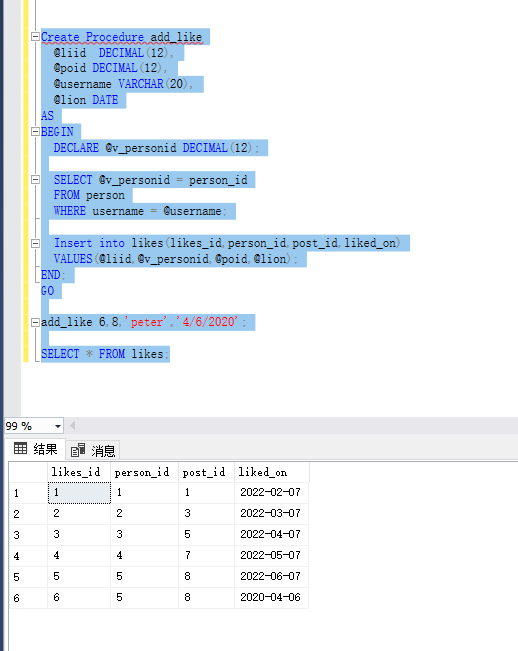
VALUES(@liid,@v\_personid,@poid,@lion);

END;

GO

add\_like 6,8,'peter','4/6/2020';

SELECT \* FROM likes;



1. Create a reusable stored procedure named “delete\_person” that takes only one parameter, the username of a person, and deletes all record of that person from the database. This means deleting all of a person’s posts, likes, and the Person record itself. Execute the procedure to delete a person of your choosing (make sure the person has at least one post and at least one like). List out all three tables to show that all record of the person is gone.

Create Procedure delete\_person

@username VARCHAR(20)

AS

BEGIN

DELETE FROM post

Where person\_id IN (SELECT person\_id

FROM person

Where username = @username);

DELETE FROM likes

Where person\_id IN (SELECT person\_id

FROM person

Where username = @username);

DELETE FROM person

Where username = @username;

END;

GO

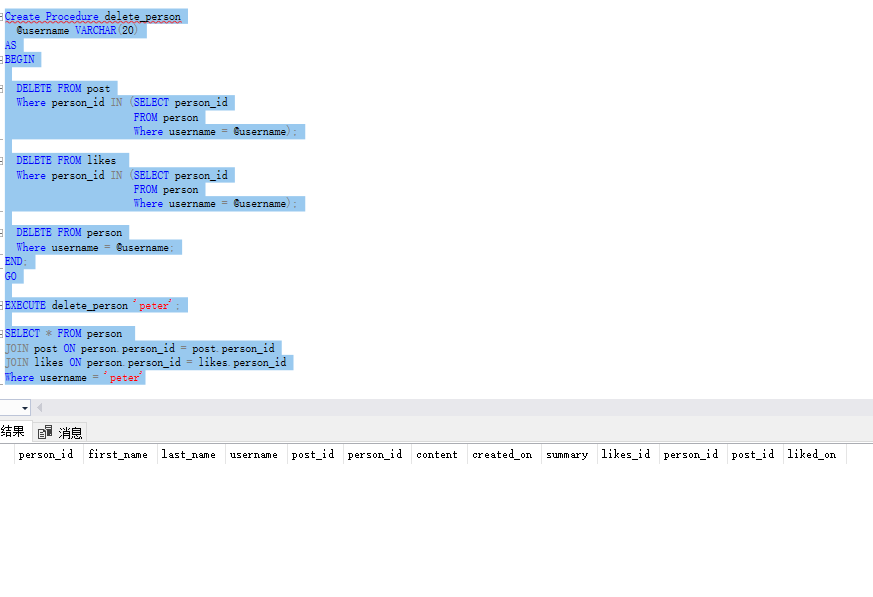
EXECUTE delete\_person 'peter';

SELECT \* FROM person

JOIN post ON person.person\_id = post.person\_id

JOIN likes ON person.person\_id = likes.person\_id

Where username = 'peter'



**Section TWO – Triggers**

1. One practical use of a trigger is validation within a single table (that is, the validation can be performed by using columns in the table being modified). Create a trigger that validates that the summary is being inserted correctly, that is, that the summary is actually the first 12 characters of the content followed by “…”. The trigger should reject an insert that does not have a valid summary value. Verify the trigger works by issuing two insert commands – one with a correct summary, and one with an incorrect summary. List out the Post table after the inserts to show one insert was blocked and the other succeeded.

Create TRIGGER checksummary

ON post AFTER INSERT,UPDATE

AS

BEGIN

DECLARE @sum VARCHAR(15);

SET @sum = (SELECT INSERTED.summary FROM INSERTED);

DECLARE @post VARCHAR(225)

SET @post= (SELECT INSERTED.content FROM INSERTED);

IF (@sum != CONCAT(SUBSTRING(@post,1,12),'...'))

BEGIN

ROLLBACK;

RAISERROR('SUMMARY NOT RIGHT FORMAT!',14,1);

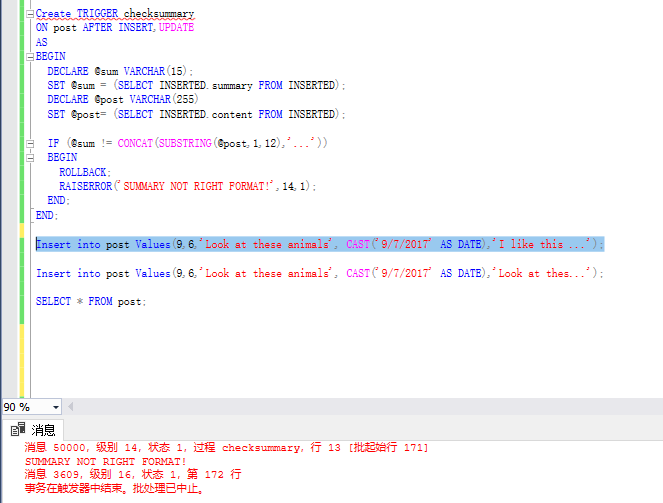
END;

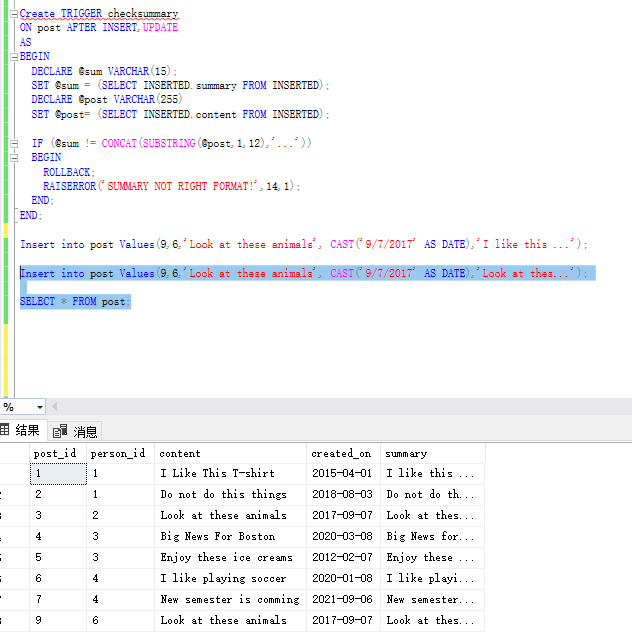
END;

Insert into post Values(9,6,'Look at these animals', CAST('9/7/2017' AS DATE),'I like this ...');

Insert into post Values(9,6,'Look at these animals', CAST('9/7/2017' AS DATE),'Look at thes...');

SELECT \* FROM post





1. Another practical use of a trigger is cross-table validation (that is, the validation needs columns from at least one table external to the table being updated). Create a trigger that blocks a “like” from being inserted if its “liked\_on” date is before the post’s “created\_on” date. Verify the trigger works by inserting two “likes” – one that passes this validation, and one that does not. List out the Likes table after the inserts to show one insert was blocked and the other succeeded.

Create TRIGGER checklike

ON likes AFTER INSERT,UPDATE

AS

BEGIN

DECLARE @likeon DATE;

DECLARE @createdon DATE;

SELECT @likeon = INSERTED.liked\_on FROM INSERTED;

SELECT @createdon = created\_on

FROM post

JOIN INSERTED ON INSERTED.post\_id = post.post\_id;

IF @likeon<@createdon

BEGIN

ROLLBACK;

RAISERROR('The like on date is earlyer than created on date!',14,1);

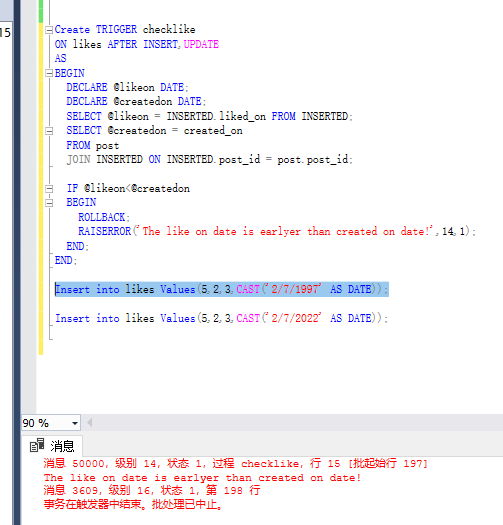
END;

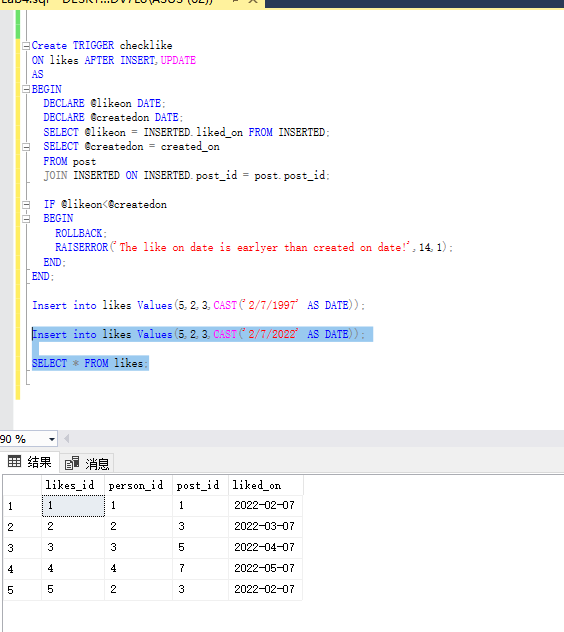
END;

Insert into likes Values(5,2,3,CAST('2/7/1997' AS DATE));

Insert into likes Values(5,2,3,CAST('2/7/2022' AS DATE));

SELECT \* FROM likes;





1. Another practical use of trigger is to maintain a history of values as they change. Create a table named post\_content\_history that is used to record updates to the content of a post, then create a trigger that keeps this table up-to-date when updates happen to post contents. Verify the trigger works by updating a post’s content, then listing out the post\_content\_history table (which should have a record of the update).

Create Table post\_content\_history(

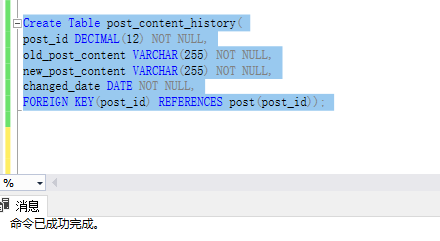
post\_id DECIMAL(12) NOT NULL,

old\_post\_content VARCHAR(255) NOT NULL,

new\_post\_content VARCHAR(255) NOT NULL,

changed\_date DATE NOT NULL,

FOREIGN KEY(post\_id) REFERENCES post(post\_id));



Create TRIGGER updatehis

ON post AFTER UPDATE

AS

BEGIN

DECLARE @oldcontent VARCHAR(255);

SELECT @oldcontent = post.content

FROM post,INSERTED

WHERE post.post\_id = INSERTED.post\_id;

DECLARE @newcontent VARCHAR(255) = (SELECT content FROM INSERTED);

DECLARE @poid DECIMAL(12) = (SELECT post\_id FROM INSERTED);

IF @oldcontent <> @newcontent

BEGIN

Insert into post\_content\_history(post\_id,old\_post\_content,new\_post\_content,changed\_date)

VALUES(@poid,@oldcontent,@newcontent,GETDATE());

END;

END;

GO

UPDATE post

SET content='Small News for Boston', summary='Small News f...'

Where post\_id = 4;

SELECT \* FROM post\_content\_history

JOIN post on post.post\_id = post\_content\_history.post\_id;

