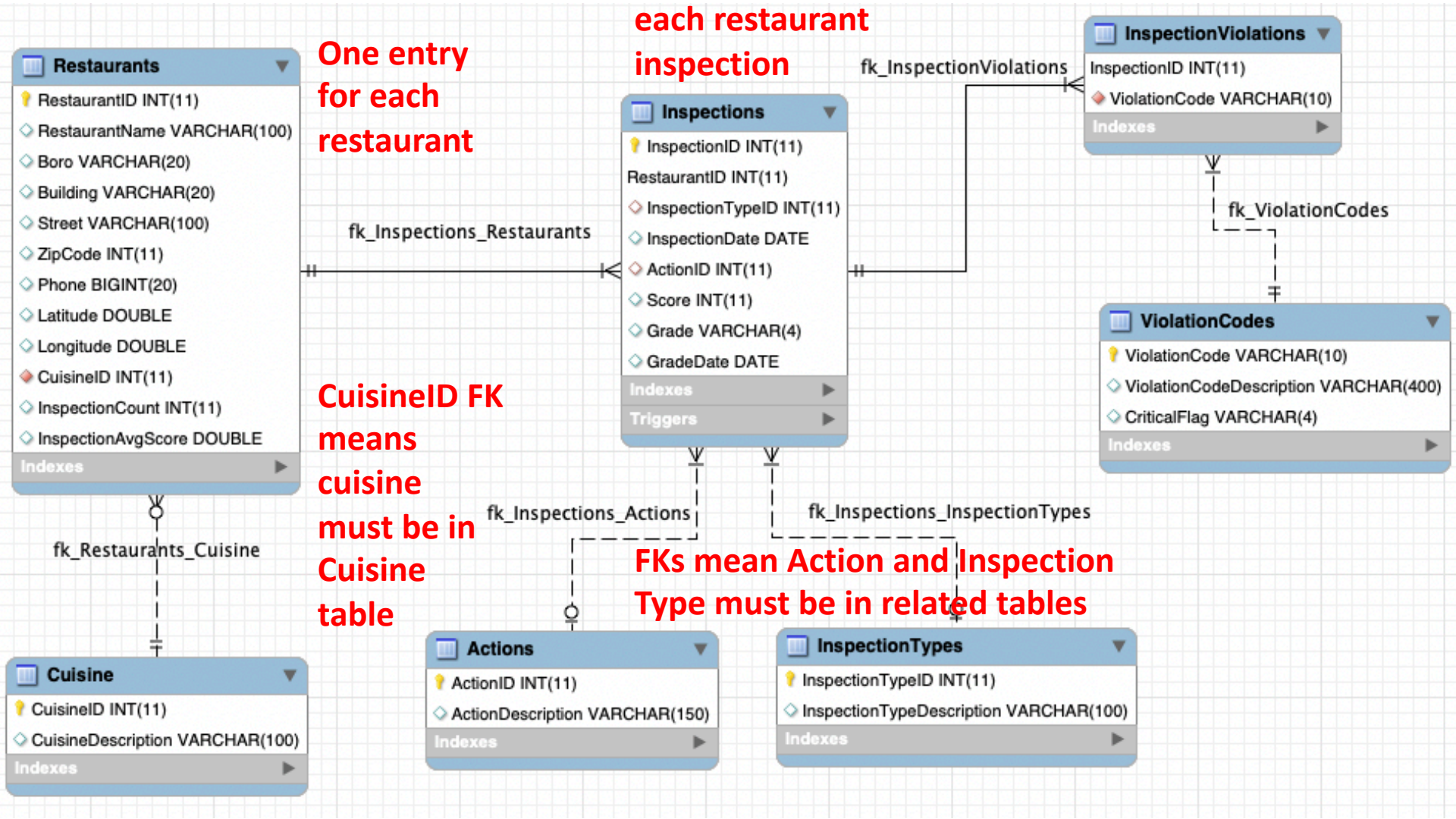
CS 61:
Database Systems

Advanced SQL

Adapted from https://www.mysqltutorial.org/ unless otherwise noted

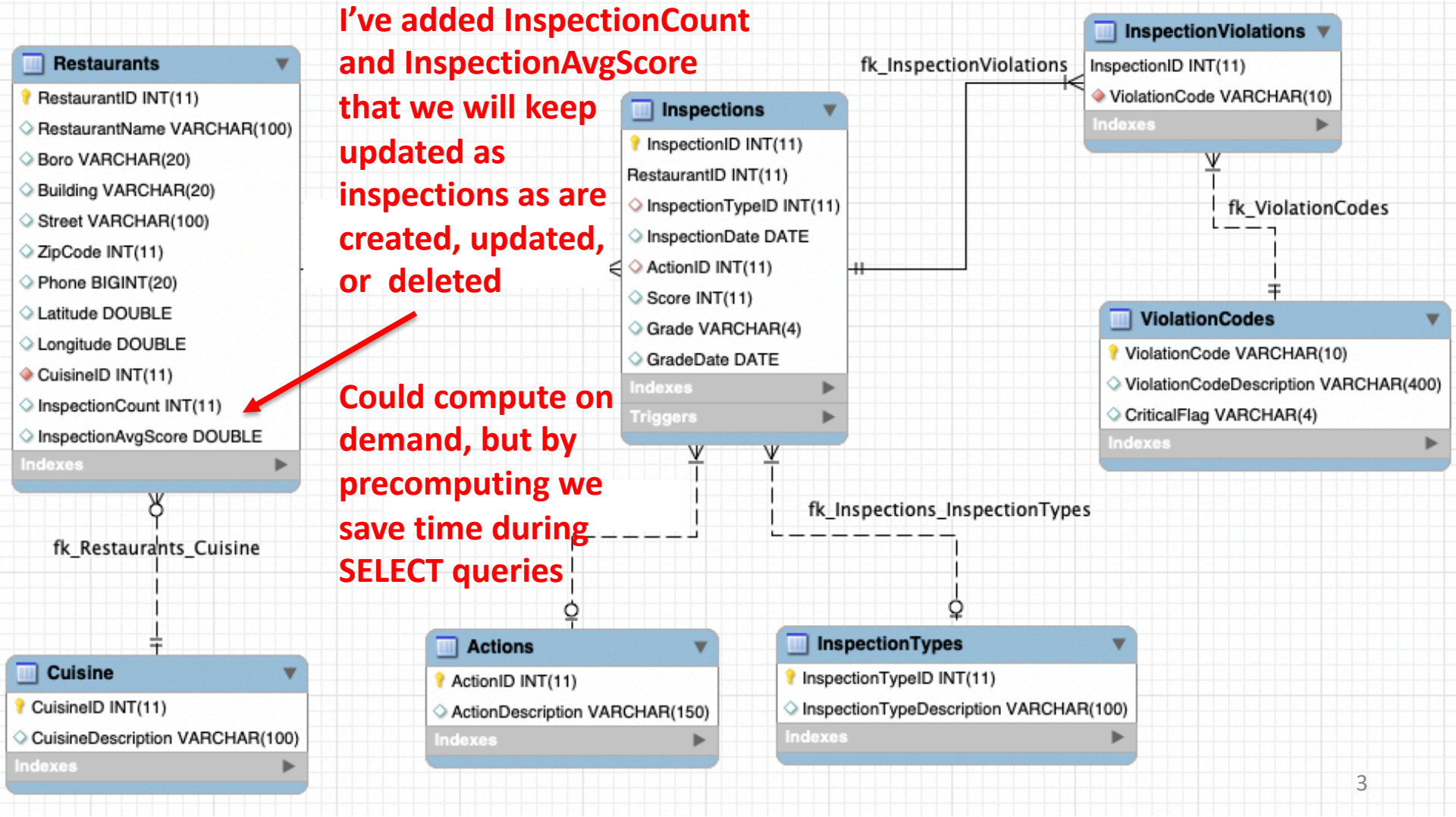
Review: database schema has tables for Restaurants and Inspections (and others)

use nyc_inspections;



Added two columns to Restaurants that we will keep updated as inspections change

use nyc_inspections;



Agenda

1. Stored procedures and functions
2. Triggers

NOTE: we can use variables in SQL, either by setting values directly or via query

Set variable value directly

```
use nyc_inspections;

SET @RestaurantID = 1111;
SELECT @RestaurantID;
```

100% 1:6

Result Grid Filter Rows: Search

@RestaurantID
1111

Set value in query

```
SELECT RestaurantName, Boro INTO @RestaurantName, @Boro
FROM Restaurants
WHERE RestaurantID = @RetaurantID;

SELECT @RestaurantName, @Boro;
```

RestaurantID from previous query (value 1111)

100% 1:12

Result Grid Filter Rows: Search Export:

@RestaurantName	@Boro
Tim's Tasty Treats'	Manhattan

- No need to declare variable or type
- Format: @varname
- To see value use variable in SELECT statement

- Use SELECT columns INTO variables
- Can have multiple variables, but only one row
- Use LIMIT 1 if query would return more than one row

Stored procedures and functions allow us to store business logic in the database

In the "bad old days" we embedded SQL directly into our application programs. This caused problems:

- What if multiple applications access the same database, how do we make sure they both implement the same business logic?
- How do we keep multiple applications following the same rules when changes occur?

Downsides:
If you use a lot of stored procedures and functions, tends to increase memory utilization

Stored procedures and functions allow us to move some business logic into the database itself

Also difficult to debug (no means to stop query execution and examine state)

- Now changes made in a single place
- Can make changes to logic and may not break applications

SQL is reasonably consistent across database vendors, but functions and stored procedures tend to be vendor-specific (our focus is MySQL)

Stored procedures allow us to save one or more SQL statements

Consider the following query

```
use nyc_inspections;

-- consider this query
SELECT * FROM Restaurants WHERE boro = 'Manhattan';
```

When you run this query from MySQL Workbench, database runs it and returns results as shown

- **If you run this query a lot, you might want to save it so you can easily run it again**
- **If you save it, the database can compile it for *possibly* slightly faster execution**
- **Could use a view, but views have trouble with updates and deletes**
- **Stored procedures are *far* more powerful than views**

100% 1:12

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

RestaurantID	RestaurantName	Boro	Building	Street	Zip
1111	Tim's Tasty Treats	Manhattan	NULL	NULL	NUL
30191841	DJ REYNOLDS PUB AND RESTAU…	Manhattan	351	WEST 57 STREET	10
40359480	1 EAST 66TH STREET KITCHEN	Manhattan	1	EAST 66 STREET	10
40362264	P & S DELI GROCERY	Manhattan	730	COLUMBUS AVENUE	10
40362274	ANGELIKA FILM CENTER	Manhattan	18	WEST HOUSTON STR…	10
40362715	THE COUNTRY CAFE	Manhattan	60	WALL STREET	10
40363298	CAFE METRO	Manhattan	625	8 AVENUE	10
40363426	LEXLER DELI	Manhattan	405	LEXINGTON AVENUE	10
40363630	LORENZO & MARIA'S KITCHEN	Manhattan	1418	THIRD AVENUE	10

To create a stored procedure in MySQL, first change the delimiter

```
6   -- save query as stored procedure
7   DELIMITER $$
8   CREATE PROCEDURE GetManhattanRestaurants()
9   BEGIN
10      SELECT * FROM Restaurants WHERE boro = 'Manhattan';
11  END$$
12  DELIMITER ;
13
14  -- call stored procedure
15  CALL GetManhattanRestaurants();
```

100% 13:15

Result Grid Filter Rows: Search Export: Fetch rows:

RestaurantID	RestaurantName	Boro	Building	Street	ZipCode	Pho
1111	Tim's Tasty Treats	Manhattan	NULL	NULL	NULL	NULL
30191841	DJ REYNOLDS PUB AND RESTAU…	Manhattan	351	WEST 57 STREET	10019	212
40359480	1 EAST 66TH STREET KITCHEN	Manhattan	1	EAST 66 STREET	10065	212
40362264	P & S DELI GROCERY	Manhattan	730	COLUMBUS AVENUE	10025	212
40362274	ANGELIKA FILM CENTER	Manhattan	18	WEST HOUSTON STR…	10012	212
40362715	THE COUNTRY CAFE	Manhattan	60	WALL STREET	10005	347
40363298	CAFE METRO	Manhattan	625	8 AVENUE	10018	212
40363426	LEXLER DELI	Manhattan	405	LEXINGTON AVENUE	10174	212
40363630	LORENZO & MARIA'S KITCHEN	Manhattan	1418	THIRD AVENUE	10028	212

- **A stored procedure may have many commands separated by ;**
- **Temporarily change delimiter to something else ($$, //, etc) so MySQL knows the function is not done until it encounters the delimiter again**
- **Change delimiter back to ; at end**

Then add your SQL, and change the delimiter back to a semicolon

```
6   -- save query as stored procedure
7   DELIMITER $$
8   CREATE PROCEDURE GetManhattanRestaurants()
9   BEGIN
10      SELECT * FROM Restaurants WHERE boro = 'Manhattan';
11  END$$
12  DELIMITER ;
13
14  -- call stored procedure
15  CALL GetManhattanRestaurants();
```

Create stored procedure and give it a name

- **Can have several SQL commands between BEGIN and END statements**
- **Can call other stored procedures**

Change command delimiter back to semicolon

Procedure stored as part of database

Use CALL to execute stored procedure

100% 13:15

Same results as executing from MySQL Workbench directly

Result Grid Filter Rows: Search Export: Fetch rows:

RestaurantID	RestaurantName	Boro	Building	Street	ZipCode	Pho
1111	Tim's Tasty Treats	Manhattan	NULL	NULL	NULL	NULL
30191841	DJ REYNOLDS PUB AND RESTAU…	Manhattan	351	WEST 57 STREET	10019	212
40359480	1 EAST 66TH STREET KITCHEN	Manhattan	1	EAST 66 STREET	10065	212
40362264	P & S DELI GROCERY	Manhattan	730	COLUMBUS AVENUE	10025	212
40362274	ANGELIKA FILM CENTER	Manhattan	18	WEST HOUSTON STR…	10012	212
40362715	THE COUNTRY CAFE	Manhattan	60	WALL STREET	10005	347
40363298	CAFE METRO	Manhattan	625	8 AVENUE	10018	212
40363426	LEXLER DELI	Manhattan	405	LEXINGTON AVENUE	10174	212
40363630	LORENZO & MARIA'S KITCHEN	Manhattan	1418	THIRD AVENUE	10028	212

Call your stored procedure using the CALL command

```
6   -- save query as stored procedure
7   DELIMITER $$
8   CREATE PROCEDURE GetManhattanRestaurants()
9   BEGIN
10      SELECT * FROM Restaurants WHERE boro = 'Manhattan';
11  END$$
12  DELIMITER ;
13
14  -- call stored procedure
15  CALL GetManhattanRestaurants();
```

100% 13:15

Banks love stored procedures

- **Consistent business logic**
- **Secure – can control access**

On first call, MySQL looks up procedure name in the database catalog, compiles the code, places it in cache memory, and executes code

On subsequent calls, execute from cache

Multiple stored procedures in cache can use up memory quickly!

Each database user has its own cache!

Result Grid Filter Rows: Search Export: Fetch rows:

RestaurantID	RestaurantName	Boro	Building	Street	ZipCode	Pho
1111	Tim's Tasty Treats	Manhattan	NULL	NULL	NULL	NULL
30191841	DJ REYNOLDS PUB AND RESTAU…	Manhattan	351	WEST 57 STREET	10019	212
40359480	1 EAST 66TH STREET KITCHEN	Manhattan	1	EAST 66 STREET	10065	212
40362264	P & S DELI GROCERY	Manhattan	730	COLUMBUS AVENUE	10025	212
40362274	ANGELIKA FILM CENTER	Manhattan	18	WEST HOUSTON STR…	10012	212
40362715	THE COUNTRY CAFE	Manhattan	60	WALL STREET	10005	347
40363298	CAFE METRO	Manhattan	625	8 AVENUE	10018	212
40363426	LEXLER DELI	Manhattan	405	LEXINGTON AVENUE	10174	212
40363630	LORENZO & MARIA'S KITCHEN	Manhattan	1418	THIRD AVENUE	10028	212

Stored procedures can take input and output variables (and input/output)

```
20    DELIMITER $$
21    CREATE PROCEDURE GetRestaurantsByBoro(
22        IN BoroName VARCHAR(20),
23        OUT RestaurantCount INT)
24    BEGIN
25        SELECT * FROM Restaurants WHERE boro = BoroName;
26        SELECT count(*) INTO RestaurantCount
27            FROM Restaurants WHERE boro = BoroName;
28    END$$
29    DELIMITER ;
30
31    CALL GetRestaurantsByBoro('Manhattan',@BoroCount);
32    SELECT @BoroCount;
33
```

100% 15:31

Parameters

- **Can have multiple params**
- **Give name and domain**
- **IN – input, value not changed inside stored procedure**
- **OUT – output, value returned**
- **INOUT – input and output variable**

Result Grid Filter Rows: Search Export: Fetch rows:

RestaurantID	RestaurantName	Boro	Building	Street	ZipCode
30191841	DJ REYNOLDS PUB AND RESTAU...	Manhattan	351	WEST 57 STREET	10019
40359480	1 EAST 66TH STREET KITCHEN	Manhattan	1	EAST 66 STREET	10065
40362264	P & S DELI GROCERY	Manhattan	730	COLUMBUS AVENUE	10025
40362274	ANGELIKA FILM CENTER	Manhattan	18	WEST HOUSTON STR...	10012
40362715	THE COUNTRY CAFE	Manhattan	60	WALL STREET	10005
40363298	CAFE METRO	Manhattan	625	8 AVENUE	10018
40363426	LEXLER DELI	Manhattan	405	LEXINGTON AVENUE	10174

Stored procedures can take input and output variables (and input/output!)

```
20    DELIMITER $$
21    CREATE PROCEDURE GetRestaurantsByBoro(
22        IN BoroName VARCHAR(20),
23        OUT RestaurantCount INT)
24    BEGIN
25        SELECT * FROM Restaurants WHERE boro = BoroName;
26        SELECT count(*) INTO RestaurantCount
27            FROM Restaurants WHERE boro = BoroName;
28    END$$
29    DELIMITER ;
30
31    CALL GetRestaurantsByBoro('Manhattan',@BoroCount);
32    SELECT @BoroCount;
33
```

100% 15:31

- **This stored procedure takes BoroName as input, returns the number of Restaurants in the boro (10,651) in @BoroCount**
- **Also returns table of matching restaurants (as shown)**
- **To not return table, comment out first SELECT**
- **Can see value of @BoroCount with SELECT @BoroCount**

Result Grid Filter Rows: Search Export: Fetch rows:

RestaurantID	RestaurantName	Boro	Building	Street	ZipCode
30191841	DJ REYNOLDS PUB AND RESTAU...	Manhattan	351	WEST 57 STREET	10019
40359480	1 EAST 66TH STREET KITCHEN	Manhattan	1	EAST 66 STREET	10065
40362264	P & S DELI GROCERY	Manhattan	730	COLUMBUS AVENUE	10025
40362274	ANGELIKA FILM CENTER	Manhattan	18	WEST HOUSTON STR...	10012
40362715	THE COUNTRY CAFE	Manhattan	60	WALL STREET	10005
40363298	CAFE METRO	Manhattan	625	8 AVENUE	10018
40363426	LEXLER DELI	Manhattan	405	LEXINGTON AVENUE	10174

Stored procedures also have statements like a traditional programming language

Local variables

- Can declare local variables in stored procedures
- Cursors to get a results set (can iterate over)

Flow control

- IF THEN ELSE
- CASE
- LOOP
- WHILE
- LEAVE (exits stored procedure)
- Structured error handling

We just scratched the surface today

- Stored procedures are not as capable as a traditional programming language
- But more capable than standard SQL

Practice

use nyc_inspections;

1. Create a stored procedure to return the min, max, avg, and count of inspection scores for a given restaurant ID
   - Hint, you'll need IN and OUT variables

2. Test your procedure on Morris Park Bake Shop at 1007 Morris Park Avenue

3. Double check your results are accurate!

Stored functions are like stored procedures but return one value

- **Functions return one value**
- **Can be used anywhere a SQL expression can be used**
- **Can have parameters like stored procedures, but can only be IN**

```
40  CREATE FUNCTION IsBoro(
41      BoroName VARCHAR(20))
42  RETURNS Boolean
43  DETERMINISTIC
44  BEGIN
45      IF BoroName IN ('Manhattan','Brooklyn','Queens','Bronx','Staten Island') THEN
46          RETURN 1;
47      END IF;
48      RETURN 0;
49  END$$
50  DELIMITER ;
51
52  SELECT IsBoro('Queens') AS Queens, IsBoro('New Hampshire') AS NH;
53
54
```

100% 14:52

Result Grid Filter Rows: Search Export:

Queens	NH
1	0

Stored functions are like stored procedures but return one value

```
40  CREATE FUNCTION IsBoro(
41      BoroName VARCHAR(20))
42  RETURNS Boolean
43  DETERMINISTIC
44  BEGIN
45      IF BoroName IN ('Manhattan','Brooklyn','Queens','Bronx','Staten Island') THEN
46          RETURN 1;
47      END IF;
48      RETURN 0;
49  END$$
50  DELIMITER ;
51
52  SELECT IsBoro('Queens') AS Queens, IsBoro('New Hampshire') AS NH;
53
54
```

DETERMINISTIC means it will always return the same value for the same input

- **Allows database to cache results knowing they won't change**
- **"Assessment of the nature of a routine is based on the "honesty" of the creator"[1]**
- **Default is NOT DETERMINISTIC**

Must return a value in RETURN statement

100% 14:52

Result Grid Filter Rows: Search Export:

Queens	NH
1	0

[1] https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html

Practice

use nyc_inspections;

1. Create a function that classifies restaurants based on how many times they have been inspected. Input: number of inspection scores. Return:
   - ‘Low’ if fewer than 7 scores
   - ‘Intermediate’ if between 7 and 12 scores
   - ‘High’ if more than 12 scores

2. Use your function in a SELECT command to return each RestaurantName and its inspection classification

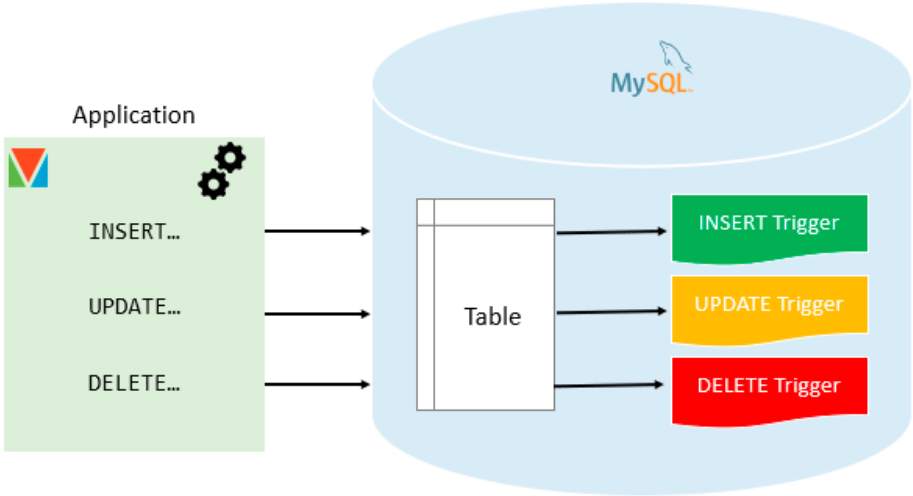
Agenda

1. Stored procedures and functions
2. Triggers

Trigger fire in response to an event such as an INSERT, UPDATE, or DELETE on a table

A trigger is a stored program invoked automatically before or after an event such as:

- INSERT
- UPDATE
- DELETE



MySQL only supports row-level triggers

- If 100 rows inserted, updated, or deleted, trigger fires 100 times
- Other databases have statement-level triggers that fire once per statement

Like most things, triggers have pros and cons

Pros

- Triggers provide another way to check the integrity of data
- Triggers give an alternative way to run scheduled tasks:
  - No need to wait for scheduled cron jobs to run
  - Triggers are invoked automatically before or after a change is made to the data in a table
- Triggers can be useful for auditing the data changes in tables
  - Make an entry into an audit table when data is added, changed, or deleted

Cons

- For simple validations, easier to use NOT NULL, UNIQUE, CHECK and FOREIGN KEY constraints
- Can be difficult to troubleshoot
  - Execute automatically in the database
  - May not invisible to client applications
- May increase processing overhead

Create trigger on Inspection table INSERT to update statistics on Restaurant table

Goal: Keep avg score and count of inspections scores current in Restaurant table when Inspection table changes (e.g., if new Inspection entered, add one to count)

```
-- create trigger on insert to Inspection table to keep count and avg score updated in Restaurants table
DROP TRIGGER IF EXISTS UpdateInspectionStatsOnInsert;
DELIMITER $$
CREATE TRIGGER UpdateInspectionStatsOnInsert
    AFTER INSERT ON Inspections
    FOR EACH ROW
    BEGIN
        -- update both columns in Restaurants table
        UPDATE Restaurants SET
            InspectionAvgScore = (SELECT AVG(Score)
                FROM Inspections i WHERE i.RestaurantID = NEW.RestaurantID LIMIT 1)
            WHERE RestaurantID = NEW.RestaurantID;
        UPDATE Restaurants SET
            InspectionCount = (SELECT count(*)
                FROM Inspections i WHERE i.RestaurantID = NEW.RestaurantID)
            WHERE RestaurantID = NEW.RestaurantID;
    END$$
DELIMITER ;
```

Give trigger a name

Can operate BEFORE or AFTER an INSERT, UPDATE, or DELETE on a specified table (Inspections)

SQL commands can reference the OLD or NEW values of an attribute

Now if a new Inspection is inserted into the Inspections table, the avg score and count are updated in Restaurants table
Can do the same for UPDATES and DELETES (see today's SQL file)

Practice

use nyc_inspections;

You're wondering if someone is paying off Health Inspectors to change inspection scores. You would like to log any changes to scores made in the Inspections table

1. Create an Audit table where we can log changes, include columns for:
   - The table that was changed (here always Inspections)
   - The primary key of the row that was changed
   - The attribute that was changed (here always Scores)
   - The score value before the change (e.g., score was a 5)
   - The score value after the change (e.g., score is now a 4)
   - The user that made the change (use the USER() function)
   - The date and time the change was made (look at CURRENT_TIMESTAMP)
2. Create a trigger that fires each time any score is updated in Inspections
3. To test, update InspectionID 26070 (Morris Park Bake Shop) from a score of 5 to a score of 4
4. Check your Audit table and confirm this change was logged
5. Are there any advantages to logging the change with a trigger vs. writing an entry into the Audit table with a user application?