SKILLPILL DATABASE

LESSON #3

1. DDL and basic administration
2. Programming in MySQL

SAIFUL HAQUE, 14 JUL 2020

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

History

E. F. Codd of the IBM San Jose Research Laboratory proposed the relational model in the late 1960s (Codd [1970]). This work led to the prestigious ACM Turing Award to Codd in 1981 (Codd [1982]). After Codd published his original paper, several research projects were formed with the goal of constructing practical relational database systems, including System R at the IBM San Jose Research Laboratory, Ingres at the University of California at Berkeley, and Query-by-Example at the IBM T. J. Watson Research Center.

The SQL programming language was first developed in the 1970s by IBM researchers Raymond Boyce and Donald Chamberlin. It was proposed under the name “SEQUEL”, which stands for **S**tructured **E**nglish **Q**uery **L**anguage.

How to pronounce?

SQL is an acronym for Structured Query Language. So therefore, SQL would be pronounced as “ess-que-ell”. And the ISO/IEC standard also uses the “S-Q-L” pronunciation (for example, “…an SQL-implementation” as opposed to “…a SQL-implementation”). Also, from the MySQL website, The official way to pronounce “MySQL” is “My Ess Que Ell” (not “my sequel”).

Despite.. some may still call it “SEQUEL” .. which is OK!

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Standards and versions

In 1986, the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) published an SQL standard, called SQL-86. ANSI Published an extended standard for SQL, SQL-89, in 1989. The next version of the standard was SQL-92 standard, followed by SQL:1999, SQL:2003, SQL:2006, SQL:2008……SQL:2016 or ISO/IEC 9075:2016

SQL-86 SQL-89 SQL-92 SQL:1999 SQL:2003 SQL:2006 SQL:2008 SQL:2011 SQL:2016

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Language	Command List	Remarks
DDL (Data Definition Language)	• CREATE • DROP • ALTER • RENAME • TRUNCATE	**Please note SQL is not case sensitive but Data is case sensitive** **Used for schema definition** **Data Definition Language, DDL, is the part of SQL that allows a database user to create and restructure database objects, such as the creation or the deletion or alteration of a table.**
DML (Data Manipulation Language)	• SELECT • INSERT • UPDATE • DELETE	**Data Manipulation Language, DML, is the part of SQL used to manipulate data within objects of a relational database.**
DCL (Data Control Language)	• GRANT • REVOKE	**Data control commands in SQL allow you to control access to data within the database. These DCL commands are normally used to create objects related to user access and also control the distribution of privileges among users.**
TCL (Transaction Control Language)	• START TRANSACTION • COMMIT • ROLLBACK	**Transactional control commands that allow the user to manage database transactions.**
Administration	• mysqldump	**mysqldump -u root -p skillpilldl company > company.sql** **mysql> source /home/s/saiful-haque/company.sql;**

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Basic Domain (data) types in SQL

- **char(n).** Fixed length character string, with user-specified length *n.*
- **varchar(n).** Variable length character strings, with user-specified maximum length *n.*
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **smallint.** Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point. (ex., **numeric**(3,1), allows 44.5 to be stores exactly, but not 444.5 or 0.32)
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

The CREATE DATABASE statement is used to create a new SQL database.

Syntax:
CREATE DATABASE <databasename>;

Example:
CREATE DATABASE <spl3+your firstname>;
CREATE DATABASE IF NOT EXISTS <spl3+your firstname>;

The DROP DATABASE statement is used to drop an existing SQL database.

Syntax:
DROP DATABASE <databasename>;
Example:
DROP DATABASE skillpillldl
DROP DATABASE IF EXISTS skillpillldl;

Connect to server hosting MySQL server:
ssh saiful-haque@skillpills <enter> [replace saiful-haque with yours]
Type AD password <enter>
Connect to MySQL DB server: mysql -u root -p <enter>
Password = skillpills <enter>

To select the database for SQL command

Syntax:
USE <databasename>;

To know which database in currently connected

Syntax:
SELECT database();

To know warnings/errors

Syntax:
SHOW warnings;
SHOW errors;

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Basic Schema Definition

We define an SQL relation by using the create table command.

Create Table Construct

- An SQL relation is defined using the **create table** command:

  create table *r* (A_1 D_1, A_2 D_2, ..., A_n D_n,
  (integrity-constraint$_1$),
  ...,
  (integrity-constraint$_k$))

  - *r* is the name of the relation
  - each A_i is an attribute name in the schema of relation *r*
  - D_i is the data type of values in the domain of attribute A_i
- Example:

```
create table instructor (
        ID          char(5),
        name        varchar(20),
        dept_name   varchar(20),
        salary      numeric(8,2));
```

The following constraints are commonly used in SQL:

NOT NULL - Ensures that a column cannot have a NULL value
UNIQUE - Ensures that all values in a column are different
PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table.
FOREIGN KEY - Uniquely identifies a row/record in another table
CHECK - Ensures that all values in a column satisfies a specific condition
DEFAULT - Sets a default value for a column when no value is specified
INDEX - Used to create and retrieve data from the database very quickly

To see table definition

Syntax:
SHOW COLUMNS FROM <tablename>;
SHOW CREATE TABLE <tablename>;

SKILLPILL DATABASE – LESSON #3
DDL AND BASIC ADMINISTRATION

Integrity Constraints in Create Table

- **not null**
- **primary key** (A_1, ..., A_n)
- **foreign key** (A_m, ..., A_n) **references** *r*

Example #2:

create table instructor(
ID **char**(5) **not null,**
name **varchar**(20) **not null,**
dept_name **varchar**(20)**,**
salary **numeric**(8,2)**,**
primary key (*ID*)**,**
foreign key(*dept_name*) **references** *department*(*dept_name*))**;**

Example #1:

create table *department(*
dept_ID **char(5) not null**,
dept_name **varchar(20) not null**,
primary key(*dept_ID*),
unique(*dept_name*));

primary key declaration on an attribute automatically ensures **not null**

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

And a Few More Relation Definitions

How to determine how many tables are the best design?

1. Relation 1-1, 1-M, M-M, and M-1
2. Cardinality 0..*,1..1
3. Normalization – 1NF (each row unique), 2NF (1NF+Column unique), 3NF (2NF+ reduce redundancy)…

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

And a Few More Relation Definitions

- **create table** *student* (
  ID **varchar**(5),
  name **varchar**(20) not null,
  dept_name **varchar**(20),
  tot_cred **numeric**(3,0),
  primary key *(ID),*
  foreign key *(dept_name*) **references** *department* (*dept_name*));
- **create table** *course* (
  course_id **varchar**(8),
  title **varchar(**50),
  dept_name **varchar**(20),
  credits **numeric**(2,0),
  semester **varchar**(6) **not null**,
  year **numeric**(4,0) **not null**,
  primary key *(course_id, semester, year),*
  foreign key *(dept_name*) **references** *department* (*dept_name*));

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

And more still

- **create table** *takes* (
  ID **varchar**(5),
  course_id **varchar**(8),
  semester **varchar**(6),
  year **numeric**(4,0),
  grade **varchar**(2),
  primary key *(ID, course_id, semester, year)* ,
  foreign key (*ID*) **references** *student* (*ID*),
  foreign key (*course_id, semester, year*) **references** *course*(*course_id, semester, year*));

  - Note: *semester* can be dropped from primary key above, to ensure a student cannot be registered for two semester of the same course.

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

And more still

```
create table employee(
        ID numeric(6,0),
        person_name char(20),
        street char(30),
        city char(30),
        primary key(ID));

create table company(
        company_name varchar(30),
        city char(30),
        primary key (company_name));
```

```
create table works(
        ID numeric(6,0),
        company_name varchar(30),
        salary integer,
        primary key (ID),
        foreign key (ID) references employee(ID),
        foreign key (company_name) references company(company_name));

create table manages(
        manager_iid numeric(6,0),
        foreign key (manager_iid) references employee(ID));
```

** Cardinality limit/total participation- say one instructor can advise 0 or more student but one student must have one advisor and maximum one advisor. Total participation is represented by double line or 0...* / 1..1

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Updates to tables

- **Insert**
  - INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);
  - INSERT INTO table_name VALUES (value1, value2, value3, ...);
- **Delete**
  - Remove all tuples from the *student* relation
    - **delete from** *student*
- **Drop Table**
  - **drop table** *r*
- **Alter**
  - **alter table** *r* **add** *A D*
    - where *A* is the name of the attribute to be added to relation *r* and *D* is the domain of *A.*
    - All exiting tuples in the relation are assigned *null* as the value for the new attribute.
  - **alter table** *r* **drop** *A*
    - where *A* is the name of an attribute of relation *r*
    - Dropping of attributes not supported by many databases.

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Playing with Insert, alter, and drop

insert into department(dept_ID, dept_name) values ('1201', 'Computer Science');

insert into department values ('1202', 'Chemistry');

insert into department values ('1203');

insert into department(dept_ID) values ('1204’);

Adding/ removing UNIQUE constraint

ALTER TABLE your_table ADD UNIQUE(target_column_name);

ALTER TABLE your_table DROP INDEX target_column_name;

Adding/ removing PRIMARY KEY constraint

ALTER TABLE your_table DROP PRIMARY KEY;

ALTER TABLE your_table ADD PRIMARY KEY (column_name);

Adding/ removing COLUMN

ALTER TABLE table_name DROP COLUMN column_name;

ALTER TABLE table_name ADD new_column_name column_definition

Modify COLUMN

ALTER TABLE table_name MODIFY COLUMN column_name datatype;

ALTER TABLE department MODIFY COLUMN dept_name varchar(20);

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Playing with Insert, alter, drop, and constraints

insert into instructor(ID,name,dept_name,salary) values ('1201','Saif','Computer Science', 100000);
insert into instructor(ID,name,dept_name,salary) values ('1201','Christian','Chemistry', 100000);
insert into instructor(ID,name,dept_name,salary) values ('1202','Christian','Chemistry', 100000);
insert into instructor(ID,name,dept_name,salary) values ('1203','Jeremie','Marine Science', 100000);
insert into department values ('1203', 'Marine Science');
insert into instructor(ID,name,dept_name,salary) values ('1204','James','Rocket Science', 100000);

ALTER TABLE `table_name` DROP FOREIGN KEY `id_name_fk`;
ALTER TABLE instructor DROP FOREIGN KEY '……'
(to know contraints ID, use "show create table instructor\g")
delete from instructor where ID = '1204';
ALTER TABLE instructor ADD FOREIGN KEY (dept_name) REFERENCES department(dept_name);

SKILLPILL DATABASE – LESSON #3
DDL AND BASIC ADMINISTRATION

The Rename Operation

- The SQL allows renaming relations and attributes using the **as** clause:

  old-name **as** *new-name*

- Find the names of all instructors who have a higher salary than some instructor in ‘Comp. Sci’.
  - **select distinct** *T.name*
    from *instructor* **as** *T, instructor* **as** *S*
    where *T.salary* > *S.salary* **and** *S.dept_name* = *‘Comp. Sci.’*

- Keyword **as** is optional and may be omitted

  instructor **as** *T* ≡ *instructor T*

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Deletion

- Delete all instructors

```
delete from instructor
```

- Delete all instructors from the Finance department

```
delete from instructor
where dept_name= 'Finance';
```

- Delete all tuples in the *instructor* relation for those instructors associated with a department located in the Watson building.

```
delete from instructor
where dept name in (select dept name
                    from department
                    where building = 'Watson');
```

SKILLPILL DATABASE – LESSON #3
DDL AND BASIC ADMINISTRATION

Deletion (Cont.)

- Delete all instructors whose salary is less than the average salary of instructors

```
delete from instructor
where salary < (select avg (salary)
                from instructor);
```

- Problem: as we delete tuples from deposit, the average salary changes
- Solution used in SQL:
  1. First, compute **avg** (salary) and find all tuples to delete
  2. Next, delete all tuples found above (without recomputing **avg** or retesting the tuples)

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Insertion

- Add a new tuple to *course*

```
insert into course
    values (’CS-437’, ’Database Systems’, ’Comp. Sci.’, 4);
```

- or equivalently

```
insert into course (course_id, title, dept_name, credits)
    values (’CS-437’, ’Database Systems’, ’Comp. Sci.’, 4);
```

- Add a new tuple to *student* with *tot_creds* set to null

```
insert into student
    values (’3003’, ’Green’, ’Finance’, null);
```

SKILLPILL DATABASE – LESSON #3
DDL AND BASIC ADMINISTRATION

Insertion (Cont.)

- Add all instructors to the *student* relation with tot_creds set to 0

```
insert into student
    select ID, name, dept_name, 0
    from   instructor
```

- The **select from where** statement is evaluated fully before any of its results are inserted into the relation.

Otherwise queries like

```
insert into table1 select * from table1
```

would cause problem

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Updates

- Increase salaries of instructors whose salary is over $100,000 by 3%, and all others by a 5%
  - Write two **update** statements:

```
update instructor
    set salary = salary * 1.03
    where salary > 100000;
update instructor
     set salary = salary * 1.05
     where salary <= 100000;
```

  - The order is important

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

```
insert into company values ('First Bank Corporation','Tokyo');
insert into company values ('OIST','Okinawa');
insert into company values ('USJ','OSaka');

insert into manages values (2),(3),(4),(6);

insert into works values (1,'OIST',110000);
insert into works values (2,'First Bank Corporation',90910);
insert into works values (3,'First Bank Corporation',90909);
insert into works values (4,'USJ',90909);
insert into works values (5,'USJ',90910);
```

```
insert into employee values (1,'Saif','Fuchaku','Onna');
insert into employee values (2,'Jeremie','Yomitan','Naha');
insert into employee values (3,'Christian','Chatan','Nago');
insert into employee values (4,'Tanaka','Fuchaku','Onna');
insert into employee values (5,'Higa','Yomitan','Naha');
insert into employee values (6,'Misaki','Chatan','Nago');
insert into employee values (7,'Nomura','Yomitan','Tokyo');
insert into employee values (8,'Misuzu','Chatan','Yokohama');
insert into employee values (9,'James','Yomitan','Saitama');
insert into employee values (10,'Sambe','Chatan','Noda');
```

Exercise

Give each manager of First Bank Corporation a 10 percent raise unless the salary becomes greater than $100,000; in such cases, give only a 3 percent raise.

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Exercise

Give each manager of First Bank Corporation a 10 percent raise unless the salary becomes greater than $100,000; in such cases, give only a 3 percent raise.

```
update works T
set T.salary = T.salary * 1.03
where T.ID in (select manager_iid from manages)
and T.salary *1.1 > 100000 and T.company_name = 'First Bank Corporation'
```

```
update works T
set T.salary = T.salary *1.1
where T.ID in (select manager_iid from manages)
and T.salary *1.1 <= 100000 and T.company_name = 'First Bank Corporation'
```

SKILLPILL DATABASE – LESSON #3

TCL: COMMIT AND ROLLBACK

Transactions

- ACID properties => Atomic, Consistent, Isolation, Durability
- Unit of work = Transaction
- Atomic transaction
  - either fully executed or rolled back as if it never occurred
- Isolation from concurrent transactions
- Transactions begin implicitly
  - Ended by **commit work** or **rollback work**
- But default on most databases: each SQL statement commits automatically
  - Can turn off auto commit for a session (e.g. using API)
  - In SQL:1999, can use: **begin atomic** …. **end**
    - Not supported on most databases

SKILLPILL DATABASE – LESSON #3

TCL: COMMIT AND ROLLBACK

```
CREATE TABLE animals (id int
NOT NULL AUTO_INCREMENT,
name char(30) NOT NULL,
PRIMARY KEY (id));

START TRANSACTION;
SET autocommit=0;
select @@autocommit;
INSERT INTO animals (name) VALUES('baboon');
INSERT INTO animals (name) VALUES('monkey');
INSERT INTO animals (name) VALUES('tiger');
COMMIT;
select * from animals;
START TRANSACTION;
DELETE FROM animals;
select * from animals;
ROLLBACK;
select * from animals;
```

```
INSERT INTO animals (name) VALUES('aardvark');
INSERT INTO animals (name) VALUES('baboon');
INSERT INTO animals (name) VALUES('monkey');
INSERT INTO animals (name) VALUES('tiger');
COMMIT;
TRUNCATE TABLE animals;
select * from animals;
ROLLBACK;
select * from animals;
```

SKILLPILL DATABASE – LESSON #3
DDL AND BASIC ADMINISTRATION

Large-Object Types

- Large objects (photos, videos, CAD files, etc.) are stored as a *large object*:
  - **blob**: binary large object -- object is a large collection of uninterpreted binary data (whose interpretation is left to an application outside of the database system)
  - **clob**: character large object -- object is a large collection of character data
  - When a query returns a large object, a pointer is returned rather than the large object itself.

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Privileges in SQL

- **select:** allows read access to relation,or the ability to query using the view
  - Example: grant users U_1, U_2, and U_3 **select** authorization on the *instructor* relation:

    grant select on *instructor* **to** U_1, U_2, U_3
- **insert**: the ability to insert tuples
- **update**: the ability to update using the SQL update statement
- **delete**: the ability to delete tuples.
- **all privileges**: used as a short form for all the allowable privileges

ALL PRIVILEGES- as we saw previously, this would allow a MySQL user full access to a designated database (or if no database is selected, global access across the system)

CREATE- allows them to create new tables or databases

DROP- allows them to them to delete tables or databases

DELETE- allows them to delete rows from tables

INSERT- allows them to insert rows into tables

SELECT- allows them to use the SELECT command to read through databases

UPDATE- allow them to update table rows

GRANT OPTION- allows them to grant or remove other users' privileges

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Revoking Authorization in SQL

- The **revoke** statement is used to revoke authorization.

  revoke <privilege list>

  on <relation name or view name> **from** <user list>

- Example:

  revoke select on *branch* **from** U_1, U_2, U_3

- <privilege-list> may be **all** to revoke all privileges the revokee may hold.
- If <revokee-list> includes **public,** all users lose the privilege except those granted it explicitly.
- If the same privilege was granted twice to the same user by different grantees, the user may retain the privilege after the revocation.
- All privileges that depend on the privilege being revoked are also revoked.

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Access control, GRANT and REVOKE

GRANT type_of_permission ON database_name.table_name TO ‘username’@'localhost’;

REVOKE type_of_permission ON database_name.table_name FROM ‘username’@‘localhost’;

Just as you can delete databases with DROP, you can use DROP to delete a user altogether:

DROP USER ‘username’@‘localhost’;

You can review a user’s current permissions by running the following:

SHOW GRANTS FOR <username>;

CREATE USER 'username'@'%' IDENTIFIED BY 'the_password’;

CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'password’;

create user 'saiftest'@'localhost' identified by 'Suzan=2020'

grant full root access:

GRANT ALL PRIVILEGES ON *.* TO 'username'@'%' WITH GRANT OPTION;

grant all privileges on *.* to 'saiftest'@'localhost' with GRANT OPTION;

always be sure to reload all the privileges:

FLUSH PRIVILEGES;

select current_user();

SELECT **User** FROM **mysql**. **user**

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Access control, GRANT and REVOKE

mysql -u saiftest -p

REVOKE all privileges ON *.* from 'saiftest'@'localhost’;

select current_user();

Try drop table, insert, select…

Exit and login as root

grant insert on skillpilldl.animals to 'saiftest'@'localhost’;

Exit and login as saiftest

insert into animals values('snake’);

Try select

select current_user();

SELECT **User** FROM **mysql**. **user**

SKILLPILL DATABASE – LESSON #3

DDL AND BASIC ADMINISTRATION

Database backup and restore in MySQL

shell> mysqldump [arguments] > file_name
To dump all databases, invoke mysqldump with the --all-databases option:
shell> mysqldump --all-databases > dump.sql

To dump only specific databases, name them on the command line and use the --databases option:

shell> mysqldump --databases db1 db2 db3 > dump.sql

To dump a single database, name it on the command line:
shell> mysqldump --databases test > dump.sql
In the single-database case, it is permissible to omit the --databases option:
shell> mysqldump test > dump.sql

To dump only specific tables from a database, name them on the command line following the database name:
shell> mysqldump test t1 t3 t7 > dump.sql

To restore the data:

From MySQL console:
mysql> source /home/s/saiful-haque/company.sql;

From shell:
mysql -u [user] -p [database_name] < [filename].sql

SKILLPILL DATABASE – LESSON #3

PROGRAMMING IN MYSQL

PL /SQL = Procedural Language for SQL

Why and when? How to catch and event?

```
CREATE TABLE animals (id int
NOT NULL AUTO_INCREMENT,
name char(30) NOT NULL,
PRIMARY KEY (id));

CREATE TABLE animal_count (animals int);

INSERT INTO animal_count (animals) VALUES(0);

CREATE TRIGGER increment_animal
AFTER INSERT ON animals
FOR EACH ROW
UPDATE animal_count SET animal_count.animals =
animal_count.animals+1;
```

The trigger has:

- a name (in this case increment_animal)
- a trigger time (in this case after the specified trigger event)
- a trigger event (an INSERT)
- a table with which it is associated (animals)
- a set of statements to run
  (here, just the one UPDATE statement)

```
INSERT INTO animals (name) VALUES('aardvark');
INSERT INTO animals (name) VALUES('baboon');
INSERT INTO animals (name) VALUES('monkey');
INSERT INTO animals (name) VALUES('tiger');
```

```
select * from animal_count;
```

SKILLPILL DATABASE – LESSON #3

PROGRAMMING IN MYSQL

Writing function in MySQL

Define a function(example):

```
CREATE FUNCTION hello (s CHAR(20))
    RETURNS CHAR(50) DETERMINISTIC
    RETURN CONCAT('Hello, ',s,'!');
```

Call a function(example):

```
select hello('Saif');
```

Drop a function(example):

```
drop function hello;
```

```
show function status\g
show create function hello\g
```

SKILLPILL DATABASE – LESSON #3

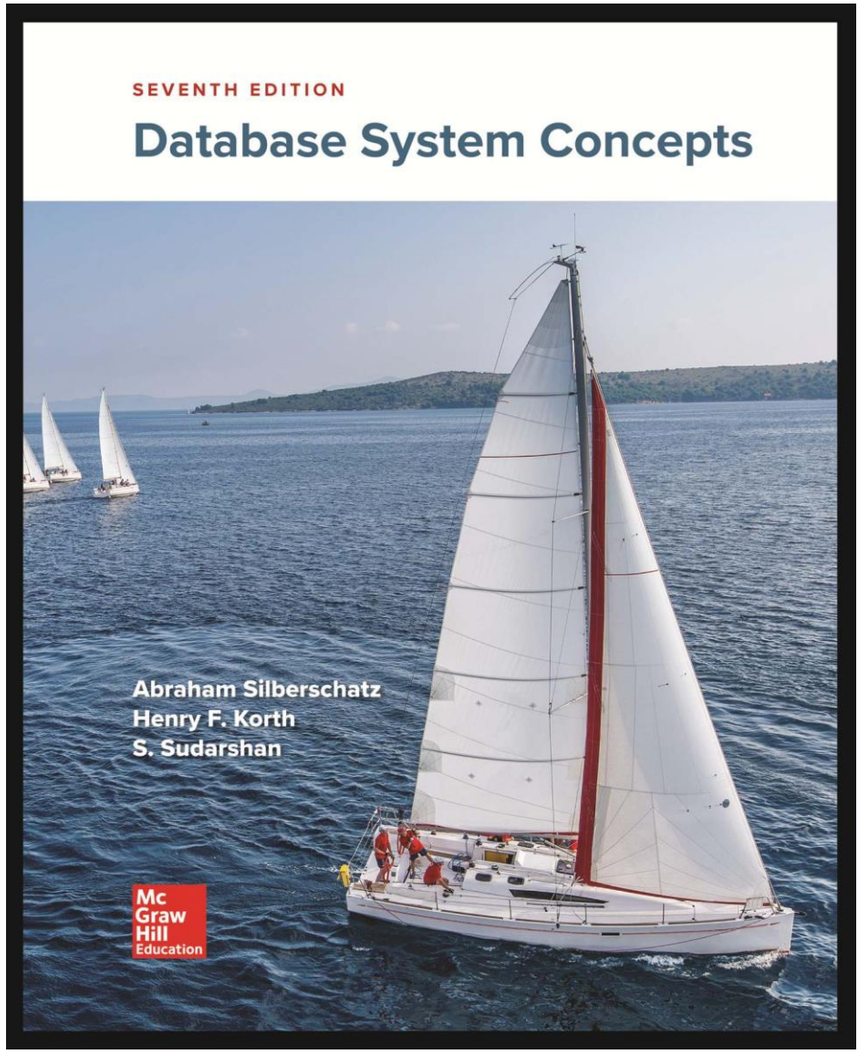
PROGRAMMING IN MYSQL

Exercise: write a MySQL function that determines the grade based on the marks obtained

```
DELIMITER //
CREATE FUNCTION getGrade(marks DECIMAL(10,2))
RETURNS VARCHAR(20) DETERMINISTIC
BEGIN
    DECLARE grade VARCHAR(20);
    IF marks >= 90 THEN SET grade ='A';
    ELSEIF (marks >= 80 AND marks <90) THEN SET grade = 'B';
    ELSEIF marks < 80 THEN SET grade = 'F';
    END IF;
RETURN (grade);
END //
DELIMITER ;
```

SKILLPILL DATABASE – LESSON #3

REFERENCE BOOK



1. DDL and basic administration
2. Programming in MySQL

THANK YOU FOR YOUR ATTENTION.

SAIFUL HAQUE, 14 JUL 2020