**Exercises**

**Rental System**

Peter runs a small car rental company with 10 cars and 5 trucks. He engages you to design a web portal to put his operation online.

For the initial phase, the web portal shall provide these basic functions:

1. Maintaining the records of the vehicles and customers.
2. Inquiring about the availability of vehicle, and
3. Reserving a vehicle for rental.

A customer record contains his/her name, address and phone number.

A vehicle, identified by the vehicle registration number, can be rented on a daily basis. The rental rate is different for different vehicles. There is a discount of 20% for rental of 7 days or more.

A customer can rental a vehicle from a start date to an end date. A special customer discount, ranging from 0-50%, can be given to preferred customers.

**Database**

The initial database contains 3 tables: vehicles, customers, and rental\_records.

The rental\_records is a *junction table* supporting many-to-many relationship

between vehicles and customers.

DROP DATABASE IF EXISTS `rental\_db`;

CREATE DATABASE `rental\_db`;

USE `rental\_db`;

-- Create `vehicles` table

DROP TABLE IF EXISTS `vehicles`;

CREATE TABLE `vehicles` (

`veh\_reg\_no` VARCHAR(8) NOT NULL,

`category` ENUM('car', 'truck') NOT NULL DEFAULT 'car',

-- Enumeration of one of the items in the list

`brand` VARCHAR(30) NOT NULL DEFAULT '',

`desc` VARCHAR(256) NOT NULL DEFAULT '',

-- desc is a keyword (for descending) and must be back-quoted

`photo` BLOB NULL, -- binary large object of up to 64KB

-- to be implemented later

`daily\_rate` DECIMAL(6,2) NOT NULL DEFAULT 9999.99,

-- set default to max value

PRIMARY KEY (`veh\_reg\_no`),

INDEX (`category`) -- Build index on this column for fast search

) ENGINE=InnoDB;

-- MySQL provides a few ENGINEs.

-- The InnoDB Engine supports foreign keys and transactions

DESC `vehicles`;

SHOW CREATE TABLE `vehicles` \G

SHOW INDEX FROM `vehicles` \G

-- Create `customers` table

DROP TABLE IF EXISTS `customers`;

CREATE TABLE `customers` (

`customer\_id` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

-- Always use INT for AUTO\_INCREMENT column to avoid run-over

`name` VARCHAR(30) NOT NULL DEFAULT '',

`address` VARCHAR(80) NOT NULL DEFAULT '',

`phone` VARCHAR(15) NOT NULL DEFAULT '',

`discount` DOUBLE NOT NULL DEFAULT 0.0,

PRIMARY KEY (`customer\_id`),

UNIQUE INDEX (`phone`), -- Build index on this unique-value column

INDEX (`name`) -- Build index on this column

) ENGINE=InnoDB;

DESC `customers`;

SHOW CREATE TABLE `customers` \G

SHOW INDEX FROM `customers` \G

-- Create `rental\_records` table

DROP TABLE IF EXISTS `rental\_records`;

CREATE TABLE `rental\_records` (

`rental\_id` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`veh\_reg\_no` VARCHAR(8) NOT NULL,

`customer\_id` INT UNSIGNED NOT NULL,

`start\_date` DATE NOT NULL DEFAULT '0000-00-00',

`end\_date` DATE NOT NULL DEFAULT '0000-00-00',

`lastUpdated` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

-- Keep the created and last updated timestamp for auditing and security

PRIMARY KEY (`rental\_id`),

FOREIGN KEY (`customer\_id`) REFERENCES `customers` (`customer\_id`)

ON DELETE RESTRICT ON UPDATE CASCADE,

-- Disallow deletion of parent record if there are matching records here

-- If parent record (customer\_id) changes, update the matching records here

FOREIGN KEY (`veh\_reg\_no`) REFERENCES `vehicles` (`veh\_reg\_no`)

ON DELETE RESTRICT ON UPDATE CASCADE

) ENGINE=InnoDB;

DESC `rental\_records`;

SHOW CREATE TABLE `rental\_records` \G

SHOW INDEX FROM `rental\_records` \G

-- Inserting test records

INSERT INTO `vehicles` VALUES

('SBA1111A', 'car', 'NISSAN SUNNY 1.6L', '4 Door Saloon, Automatic', NULL, 99.99),

('SBB2222B', 'car', 'TOYOTA ALTIS 1.6L', '4 Door Saloon, Automatic', NULL, 99.99),

('SBC3333C', 'car', 'HONDA CIVIC 1.8L', '4 Door Saloon, Automatic', NULL, 119.99),

('GA5555E', 'truck', 'NISSAN CABSTAR 3.0L', 'Lorry, Manual ', NULL, 89.99),

('GA6666F', 'truck', 'OPEL COMBO 1.6L', 'Van, Manual', NULL, 69.99);

-- No photo yet, set to NULL

SELECT \* FROM `vehicles`;

INSERT INTO `customers` VALUES

(1001, 'Tan Ah Teck', '8 Happy Ave', '88888888', 0.1),

(NULL, 'Mohammed Ali', '1 Kg Java', '99999999', 0.15),

(NULL, 'Kumar', '5 Serangoon Road', '55555555', 0),

(NULL, 'Kevin Jones', '2 Sunset boulevard', '22222222', 0.2);

SELECT \* FROM `customers`;

INSERT INTO `rental\_records` VALUES

(NULL, 'SBA1111A', 1001, '2012-01-01', '2012-01-21', NULL),

(NULL, 'SBA1111A', 1001, '2012-02-01', '2012-02-05', NULL),

(NULL, 'GA5555E', 1003, '2012-01-05', '2012-01-31', NULL),

(NULL, 'GA6666F', 1004, '2012-01-20', '2012-02-20', NULL);

SELECT \* FROM `rental\_records`;

**Exercises**

1. Customer 'Tan Ah Teck' has rented 'SBA1111A' from today for 10 days. (Hint: You need to insert a rental record. Use a SELECT subquery to get the customer\_id. Use CURDATE() (or NOW()) for today; and DATE\_ADD(CURDATE(), INTERVAL x unit) to compute a future date.)

INSERT INTO rental\_records VALUES

(NULL,

'SBA1111A',

(SELECT customer\_id FROM customers WHERE name='Tan Ah Teck'),

CURDATE(),

DATE\_ADD(CURDATE(), INTERVAL 10 DAY),

NULL);

1. Customer 'Kumar' has rented 'GA5555E' from tomorrow for 3 months.

List all rental records (start date, end date) with vehicle's registration number, brand, and customer name, sorted by vehicle's categories followed by start date.

SELECT

r.start\_date AS `Start Date`,

r.end\_date AS `End Date`,

r.veh\_reg\_no AS `Vehicle No`,

v.brand AS `Vehicle Brand`,

c.name AS `Customer Name`

FROM rental\_records AS r

INNER JOIN vehicles AS v USING (veh\_reg\_no)

INNER JOIN customers AS c USING (customer\_id)

ORDER BY v.category, start\_date;

1. List all the expired rental records (end\_date before CURDATE()).
2. List the vehicles rented out on '2012-01-10' (not available for rental), in columns of vehicle registration no, customer name, start date and end date. (Hint: the given date is in between the start\_date and end\_date.)
3. List all vehicles rented out today, in columns registration number, customer name, start date, end date.
4. Similarly, list the vehicles rented out (not available for rental) for the period from '2012-01-03' to '2012-01-18'. (Hint: start\_date is inside the range; or end\_date is inside the range; or start\_date is before the range and end\_date is beyond the range.)
5. List the vehicles (registration number, brand and description) available for rental (not rented out) on '2012-01-10' (Hint: You could use a subquery based on a earlier query).
6. Similarly, list the vehicles available for rental for the period from '2012-01-03' to '2012-01-18'.
7. Similarly, list the vehicles available for rental from today for 10 days.
8. Foreign Key Test:
   1. Try deleting a parent row with matching row(s) in child table(s), e.g., delete 'GA6666F' from vehicles table (ON DELETE RESTRICT).
   2. Try updating a parent row with matching row(s) in child table(s), e.g., rename 'GA6666F' to 'GA9999F' in vehicles table. Check the effects on the child table rental\_records (ON UPDATE CASCADE).
   3. Remove 'GA6666F' from the database (Hints: Remove it from child table rental\_records; then parent table vehicles.)
9. Payments: A rental could be paid over a number of payments (e.g., deposit, installments, full payment). Each payment is for one rental. Create a new table called payments. Need to create columns to facilitate proper audit check (such as create\_date, create\_by, last\_update\_date, last\_update\_by, etc.)

DROP TABLE IF EXISTS `payments`;

CREATE TABLE payments (

`payment\_id` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`rental\_id` INT UNSIGNED NOT NULL,

`amount` DECIMAL(8,2) NOT NULL DEFAULT 0,

`mode` ENUM('cash', 'credit card', 'check'),

`type` ENUM('deposit', 'partial', 'full') NOT NULL DEFAULT 'full',

`remark` VARCHAR(255),

`created\_date` DATETIME NOT NULL,

`created\_by` INT UNSIGNED NOT NULL, -- staff\_id

-- Use a trigger to update create\_date and create\_by automatically

`last\_updated\_date` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

-- Updated by the system automatically

`last\_updated\_by` INT UNSIGNED NOT NULL,

-- Use a trigger to update created\_by

PRIMARY KEY (`payment\_id`),

INDEX (`rental\_id`),

FOREIGN KEY (`rental\_id`) REFERENCES rental\_records (`rental\_id`)

) ENGINE=InnoDB;

DESC `payments`;

SHOW CREATE TABLE `payments` \G

SHOW INDEX FROM `payments` \G

1. Staff: Keeping track of staff serving the customers. Create a new staff table. Assume that each transaction is handled by one staff, we can add a new column called staff\_id in the rental\_records table,

DROP TABLE IF EXISTS `staff`;

CREATE TABLE `staff` (

`staff\_id` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

-- Always use INT for AUTO\_INCREMENT column to prvent run-over

`name` VARCHAR(30) NOT NULL DEFAULT '',

`title` VARCHAR(30) NOT NULL DEFAULT '',

`address` VARCHAR(80) NOT NULL DEFAULT '',

`phone` VARCHAR(15) NOT NULL DEFAULT '',

`report\_to` INT UNSIGNED NOT NULL,

-- Reports to manager staff\_id. Boss reports to himself

PRIMARY KEY (`staff\_id`),

UNIQUE INDEX (`phone`), -- Build index on this unique-value column

INDEX (`name`), -- Build index on this column

FOREIGN KEY (`report\_to`) REFERENCES `staff` (`staff\_id`)

-- Reference itself

) ENGINE=InnoDB;

DESC `staff`;

SHOW INDEX FROM `staff` \G

INSERT INTO staff VALUE (8001, 'Peter Johns', 'Managing Director', '1 Happy Ave', '12345678', 8001);

SELECT \* FROM staff;

-- Add a new column to rental\_records table

ALTER TABLE `rental\_records` ADD COLUMN `staff\_id` INT UNSIGNED NOT NULL;

-- Need to set to a valid value, before adding the foreign key

UPDATE `rental\_records` SET `staff\_id` = 8001;

ALTER TABLE `rental\_records` ADD FOREIGN KEY (`staff\_id`) REFERENCES staff (`staff\_id`)

ON DELETE RESTRICT ON UPDATE CASCADE;

SHOW CREATE TABLE `rental\_records` \G

SHOW INDEX FROM `rental\_records` \G

-- Also Add a new column to payments table

ALTER TABLE `payments` ADD COLUMN `staff\_id` INT UNSIGNED NOT NULL;

-- Need to set to a valid value, before adding the foreign key

UPDATE `payments` SET `staff\_id` = 8001;

ALTER TABLE `payments` ADD FOREIGN KEY (`staff\_id`) REFERENCES staff (`staff\_id`)

ON DELETE RESTRICT ON UPDATE CASCADE;

SHOW CREATE TABLE `payments` \G

SHOW INDEX FROM `payments` \G

**Advanced Exercises**

1. Adding Photo: We could store photo in MySQL using data type of BLOB (Binary Large Object) (up to 64KB), MEDIUMBLOB (up to 16MBytes), LONGBOLB (up to 4GBytes). For example,

-- Use function LOAD\_FILE to load a picture file into a BLOB field

UPDATE vehicles SET photo=LOAD\_FILE('d:/temp/car.jpg') WHERE veh\_reg\_no = 'SBA1111A';

SELECT \* FROM vehicles WHERE veh\_reg\_no = 'SBA1111A' \G

You can conveniently load and view the photo via graphical tools such as MySQL Workbench. To load a image in MySQL Workbench ⇒ right-click on the cell ⇒ Load Value From File ⇒ Select the image file. To view the image ⇒ right-click on the BLOB cell ⇒ Open Value in Editor ⇒ choose "Image" pane.  
I also include a Java program for reading and writing image BLOB from/to the database, based on this example: "[TestImageBLOB.java](https://www3.ntu.edu.sg/home/ehchua/programming/sql/codes/TestImageBLOB.java.txt)".

1. VIEW: Create a VIEW called rental\_prices on the rental\_records with an additional column called price. Show all the records of the VIEW.

DROP VIEW IF EXISTS rental\_prices;

CREATE VIEW **rental\_prices**

AS

SELECT

v.veh\_reg\_no AS `Vehicle No`,

v.daily\_rate AS `Daily Rate`,

c.name AS `Customer Name`,

c.discount\*100 AS `Customer Discount (%)`,

r.start\_date AS `Start Date`,

r.end\_date AS `End Date`,

DATEDIFF(r.end\_date, r.start\_date) AS `Duration`,

-- Compute the rental price

-- Preferred customer has discount, 20% discount for 7 or more days

-- CAST the result from DOUBLE to DECIMAL(8,2)

CAST(

IF (DATEDIFF(r.end\_date, r.start\_date) < 7,

DATEDIFF(r.end\_date, r.start\_date)\*daily\_rate\*(1-discount),

DATEDIFF(r.end\_date, r.start\_date)\*daily\_rate\*(1-discount)\*0.8)

AS DECIMAL(8,2)) AS price

FROM rental\_records AS r

INNER JOIN vehicles AS v USING (veh\_reg\_no)

INNER JOIN customers AS c USING (customer\_id);

DESC `rental\_prices`;

SHOW CREATE VIEW `rental\_prices` \G

-- Try selecting all rows

SELECT \* FROM `rental\_prices`;

It is probably easier to compute the price using a program/procedure, instead of inside the view.

1. From the payments table, create a view to show the outstanding balance.
2. Define more views.
3. FUNCTION: Write a function to compute the rental price.
4. Define more procedures and functions.
5. TRIGGER: Write a trigger for the created\_date and created\_by columns of the payments table.
6. Define more triggers.
7. Implement discount on weekday (Monday to Friday, except public holiday): Need to set up a new table called public\_hoilday with columns date and description. Use function DAYOFWEEK (1=Sunday, …, 7=Saturday) to check for weekday or weekend.

-- pseudocode for calculating rental price

price = 0;

for each date from start\_date to end\_date {

if date is weekend or public\_holiday, price += daily\_rate;

else price += daily\_rate\*(1-discount);

}

if (duration >= 7) price \*= (1 - long\_duration\_discount);

price \*= (1 - perferred\_customer\_discount);

**7.2  Product Sales Database**

