

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II

Θέμα: Υλοποίηση ΒΔ και εφαρμογής διαχείρισης
ασφαλιστικών προϊόντων



Εργαστήριο 2020-2021, Τμήμα Δευτέρας
Κος Ε. Κοσμάτος

Εργασία 2
Ημ. Παράδοσης: 16 Δεκ. 2020

ΒΕΛΑΣΚΟ ΠΑΟΛΑ
ΚΟΥΜΠΟΥΡΗ ΜΑΡΓΑΡΙΤΑ

161020
18390109

Πίνακας περιεχομένων

ΣΚΟΠΟΣ ΕΡΓΑΣΙΑΣ	2
ΕΙΣΑΓΩΓΗ	2
Κώδικας/ Εντολές – Εργ. Άσκηση 1	2
ΥΛΟΠΟΙΗΣΗ ΕΡΓΑΣΙΑΣ	8
1. Triggers.....	8
Βήμα 6	8
Εντολές	8
Τεκμηρίωση	9
Αποτελέσματα - Παραδείγματα	10
2. Functions.....	13
Βήμα 7	13
Εντολές	13
Τεκμηρίωση	13
Αποτελέσματα - Παραδείγματα	14
3. Procedures	15
Βήμα 8	15
Εντολές	15
Τεκμηρίωση	16
Αποτελέσματα - Παραδείγματα	16
ΣΥΜΠΕΡΑΣΜΑΤΑ	18

ΣΚΟΠΟΣ ΕΡΓΑΣΙΑΣ

Η εκπόνηση της εργαστηριακής άσκηση 2 με τίτλο «Δημιουργία κατάλληλων triggers, procedures, functions στη Βάση Δεδομένων» αποτελεί τη συνέχεια της προηγούμενης άσκησης. Ωστόσο, ο σκοπός της είναι η εξοικείωσή μας σε καινούρια εργαλεία που παρέχει η MySQL και συγκεκριμένα στα “trigger”, “functions” και “procedures”.

ΕΙΣΑΓΩΓΗ

Στην εργαστηριακή άσκηση 1 υλοποιήσαμε την απαραίτητη βάση δεδομένων σύμφωνα με τους περιορισμούς που μας τέθηκαν.

Παρατίθεται παρακάτω ο κώδικας/εντολές από την άσκηση 1, ο οποίος είναι αναγκαίος και σε αυτή την άσκηση, καθώς η άσκηση 2 είναι η επέκτασή της.

Κώδικας/ Εντολές – Εργ. Άσκηση 1

```
DROP DATABASE IF EXISTS General_Insurance;
CREATE DATABASE General_Insurance;
USE General_Insurance;

#table customer
CREATE TABLE customer
(
    AFM int(30) not null,
    name varchar(30),
    surname varchar(30),
    address varchar(30),
    DOY varchar(4),
    phone varchar(13),
    PRIMARY KEY (AFM),
    CONSTRAINT phone_check CHECK (REGEXP_LIKE(phone, '^[+][0-9]{12}$'))
);

#table coverage ( type of service coverage )
CREATE TABLE coverage
(
    coverage_code int not null auto_increment,
    coverage_name varchar(70),
    PRIMARY KEY (coverage_code)
);

#table insurance
CREATE TABLE insurance
(
    insurance_code int not null,
    insurance_name varchar(60),
    annual_cost int(5),
    min_duration int not null,
    PRIMARY KEY (insurance_code),
```

```

    CONSTRAINT cost_check CHECK (annual_cost > 0)
);

#table insurance_coverages
CREATE TABLE insurance_coverages
(
    insurance_code int not null,
    coverage_code int not null auto_increment,
    PRIMARY KEY (insurance_code, coverage_code),
    FOREIGN KEY (insurance_code) REFERENCES insurance (insurance_code),
    FOREIGN KEY (coverage_code) REFERENCES coverage (coverage_code)
);

#table contract
CREATE TABLE contract
(
    contract_code int not null,
    cost int(5),
    start_date date,
    END_date date,
    AFM int(30) not null,
    insurance_code int not null,
    PRIMARY KEY (contract_code),
    FOREIGN KEY (AFM) REFERENCES customer (AFM),
    FOREIGN KEY (insurance_code) REFERENCES insurance (insurance_code)
);

#table insurance_customer
CREATE TABLE insurance_customer
(
    insurance_code int not null,
    AFM int(30) not null,
    PRIMARY KEY (insurance_code, AFM),
    FOREIGN KEY (AFM) REFERENCES customer (AFM),
    FOREIGN KEY (insurance_code) REFERENCES insurance (insurance_code)
);

#show tables of database
show tables;

#DESCRIBE tables of General_Insurance
DESCRIBE contract;
DESCRIBE customer;
DESCRIBE insurance;
DESCRIBE coverage;
DESCRIBE insurance_coverages;
DESCRIBE insurance_customer;

SELECT * FROM customer;
#INSERT INTO customer
INSERT INTO customer (AFM, name, surname, phone, address, DOY)
VALUES
(1023452569, 'Charlie', 'Hunnam', '+602103625956', 'California', 'A'),
(1445525690, 'Henry', 'Cavill', '+902104135956', 'Chicago', 'B'),
(1785592569, 'Orlando', 'Bloom', '+302103681956', 'Athens', 'C'),
(1785527877, 'James', 'Smith', '+302103765556', 'Thessaloniki', 'D'),

```

```

(1884560847, 'Jason', 'Steven', '+302108741262', 'Kavala', 'E'),
(1542147114, 'Paola', 'Velasco', '+302114522704', 'Athens', 'A'),
(1445547451, 'Margarita', 'Koumpouri', '+302115475668', 'Athens', 'B'),
(1445477457, 'Ramez', 'Elmasri', '+662108544615', 'Arlington', 'C'),
(1024782463, 'Shamkant', 'Navathe', '+852103659569', 'Michigan', 'D'),
(1841204754, 'Maria', 'Papadopoulou', '+302117941554', 'Patras', 'A'),
(1784585254, 'Kostas', 'Pappas', '+302135874572', 'Lamia', 'B'),
(1745242426, 'Andreas', 'Dimitrios', '+302108564524', 'Volos', 'I'),
(1415541574, 'Tatiana', 'Raptis', '+302105448542', 'Thessaloniki',
'Z'),
(1124834578, 'Periklis', 'Megalos', '+302115858645', 'Volos', 'A'),
(1486542484, 'Liliana', 'Beckham', '+902138524656', 'Chicago', 'C'),
(1946923008, 'HENDERSON', 'Gibb', '+602126799446', 'California', 'C'),
(1256452238, 'Alexandra', 'Denman', '+302100585464', 'Patras', 'A'),
(1592227645, 'Thea', 'Wade', '+302118548646', 'Athens', 'B');

#INSERT INTO coverage
INSERT INTO coverage (coverage_code, coverage_name)
VALUES
    (1, 'medicine');

SELECT * FROM coverage;

INSERT INTO coverage (coverage_name)
VALUES
    ('maternity'),
    ('funeral'),
    ('accident'),
    ('pharmacy'),
    ('repair'),
    ('transplant'),
    ('vacation cover for terminal people'),
    ('property');

SELECT * FROM coverage;

SELECT * FROM insurance;

#INSERT INTO insurance
INSERT INTO insurance(insurance_code, insurance_name, annual_cost,
min_duration)
VALUES (10, 'Health', 500, 1),
    (20, 'Critical', 600, 1),
    (30, 'Home', 300, 3),
    (40, 'Car', 200, 2);
SELECT * FROM insurance;

SELECT * FROM insurance_coverages;

#INSERT INTO insurance_coverages
INSERT INTO insurance_coverages (insurance_code, coverage_code)
VALUES
    (10, 1),
    (10, 2),
    (10, 4),
    (20, 1),

```

```
(20, 3),
(30, 6),
(30, 9),
(40, 4),
(40, 6);

DELETE FROM contract;
SELECT * FROM contract;

#INSERT INTO contract  DATE (YYYY-M-D)
INSERT INTO contract
(contract_code,cost,start_date,END_date,AFM,insurance_code)
VALUES
(512, 11000, '2000-1-1', '2022-1-1', 1023452569, 10),
(513, 1800, '2006-1-1', '2009-1-1', 1445525690, 20),
(514, 4000, '2004-1-1', '2006-1-1', 1785592569, 30),
(515, 900, '2004-1-1', '2010-1-1', 1785592569, 10),
(516, 1000, '2005-1-1', '2010-1-1', 1884560847, 10),
(517, 2500, '2005-3-5', '2014-3-5', 1884560847, 20),
(518, 1800, '2008-3-6', '2020-3-6', 1884560847, 30),
(519, 2000, '2011-6-4', '2017-6-4', 1884560847, 40),
(520, 1200, '2011-6-4', '2015-6-4', 1542147114, 40),
(521, 2000, '2011-11-9', '2014-11-9', 1445547451, 10),
(522, 900, '2011-7-5', '2015-7-5', 1445477457, 30),
(523, 1500, '2012-3-8', '2015-3-8', 1024782463, 10),
(524, 1800, '2012-8-9', '2019-8-9', 1024782463, 20),
(525, 900, '2013-9-5', '2019-9-5', 1841204754, 30),
(526, 900, '2013-9-5', '2018-9-5', 1784585254, 30),
(527, 900, '2015-11-10', '2022-11-10', 1745242426, 30),
(528, 1200, '2016-12-10', '2019-12-10', 1415541574, 40),
(529, 400, '2017-4-30', '2023-4-30', 1124834578, 40),
(530, 1200, '2017-6-2', '2020-6-2', 1486542484, 40),
(531, 1500, '2017-6-3', '2025-6-3', 1946923008, 10),
(532, 4800, '2017-6-4', '2026-6-4', 1256452238, 20),
(533, 4000, '2018-9-5', '2026-9-5', 1592227645, 10),
(534, 4800, '2018-9-5', '2026-9-5', 1592227645, 20),
(535, 1800, '2018-9-5', '2024-9-5', 1592227645, 30),
(536, 1600, '2018-9-5', '2026-9-5', 1592227645, 40);

#INSERT INTO insurance_customer
INSERT INTO insurance_customer (insurance_code, AFM)
VALUES
(10, 1023452569),
(10, 1785592569),
(10, 1884560847),
(20, 1445525690),
(20, 1884560847),
(30, 1884560847),
(30, 1784585254),
(40, 1486542484),
(40, 1592227645);

#show current contents of tables
SELECT * FROM contract;
SELECT * FROM customer;
```

```
SELECT * FROM insurance;
SELECT * FROM insurance_coverages;
SELECT * FROM insurance_customer;

#updatable view (unsafe)
DROP VIEW IF EXISTS customerInfo;

CREATE VIEW customerInfo
AS SELECT name, surname, address FROM customer;

SELECT * FROM customerInfo;

UPDATE customerInfo SET address = "Dublin" WHERE name = "Ramez" AND
surname = "Elmasri";

SELECT * FROM customerInfo;
SELECT * FROM customer;

#updatable view (unsafe with certain join)
DROP VIEW IF EXISTS customerContract_view;
CREATE VIEW customerContract_view
AS SELECT customer.AFM, name, surname
FROM customer INNER JOIN contract ON customer.AFM = contract.AFM;

SELECT * FROM customerContract_view;
SELECT * FROM customer;

UPDATE customerContract_view SET name = "Stefan" WHERE AFM =
1023452569;
SELECT * FROM customerContract_view;

#updatable view (safe)
DROP VIEW IF EXISTS customers_view3;
CREATE VIEW customers_view3 (AFM, name, surname, address, DOY, phone)
AS SELECT * FROM customer
WHERE DOY in ("A", "C")
WITH CHECK OPTION;

INSERT INTO customers_view3
VALUES
    (178663326, 'John', 'Velasco', 'Athens', 'C', '+302115055293');
SELECT * FROM customers_view3;
SELECT * FROM customer;

INSERT INTO customers_view3 ( AFM, name, surname, phone, address, DOY )
VALUES
    (177452826, 'Christopher', 'Velasco', '+102115783293', 'London', 'L');

#non updatable view
DROP VIEW IF EXISTS cust_distinct_names;
```

```
CREATE VIEW cust_distinct_names(name)
AS SELECT DISTINCT name FROM customer ORDER BY name;

SELECT * FROM cust_distinct_names;
INSERT INTO cust_distinct_names VALUES ('Gerard');

# Show how many contracts have been signed for each insurance
SELECT insurance_name, count(*) AS 'contracts signed'
FROM contract INNER JOIN insurance
ON contract.insurance_code = insurance.insurance_code
GROUP BY contract.insurance_code;

# Show the clients according to the total of the contracts that they
have signed (in descending order)
SELECT DISTINCT name, surname, sum(cost) AS total
FROM customer INNER JOIN contract
ON customer.AFM = contract.AFM
GROUP BY customer.AFM
ORDER BY total DESC;
```


ΥΛΟΠΟΙΗΣΗ ΕΡΓΑΣΙΑΣ

1. Triggers

Βήμα 6

Δημιουργήστε *triggers* που θα ενημερώνουν αυτόματα το συνολικό κόστος των συμβολαίων ανά πελάτη (*cost_of_contracts*).

Εντολές

```
#*****TRIGGERS*****

ALTER TABLE customer ADD (con_sum INT(3));
UPDATE customer SET con_sum =
(SELECT IFNULL(sum(cost),0) FROM contract WHERE customer.AFM =
contract.AFM);

SELECT * FROM customer;
SELECT * FROM contract;

#*****trigger: after insert*****

DROP TRIGGER IF EXISTS contract_insert;

delimiter //
CREATE TRIGGER contract_insert
AFTER INSERT ON contract
FOR EACH ROW
BEGIN
    UPDATE customer
    SET con_sum = con_sum + NEW.cost
    WHERE customer.AFM = NEW.AFM;
END;
//
delimiter ;

INSERT INTO contract VALUES (539, 500, '2024-12-15', '2025-12-15',
178663326, 10);
INSERT INTO contract VALUES (540, 1000, '2024-12-15', '2025-12-15',
1023452569, 10);

SELECT * FROM customer;
SELECT * FROM contract;

#*****trigger: after delete*****

DROP TRIGGER IF EXISTS contract_delete;

delimiter //
CREATE TRIGGER contract_delete
```

```

AFTER DELETE on contract
FOR EACH ROW
BEGIN
    UPDATE customer
    SET con_sum = con_sum - OLD.cost
    WHERE customer.AFM = OLD.AFM;
END;
//
delimiter ;

DELETE FROM contract WHERE contract_code = 528;
SELECT * FROM customer;
SELECT * FROM contract;

#*****trigger: after update*****

DROP TRIGGER IF EXISTS contract_update;
delimiter //

CREATE TRIGGER contract_update
AFTER UPDATE on contract
FOR EACH ROW
BEGIN
    UPDATE customer SET con_sum =
    (SELECT ifnull(sum(cost),0) FROM contract WHERE customer.AFM =
contract.AFM);
END;
//
delimiter ;

UPDATE contract SET cost = 623 WHERE contract_code = 517;
SELECT * FROM customer;
SELECT * FROM contract;

#*****info about TRIGGERS*****

DESCRIBE Information_schema.TRIGGERS;
SELECT TRIGGER_NAME, EVENT_MANIPULATION, TRIGGER_SCHEMA
FROM INFORMATION_SCHEMA.TRIGGERS
WHERE TRIGGER_SCHEMA = 'general_insurance'
ORDER BY TRIGGER_NAME;

```

Τεκμηρίωση

- ✓ Το contract_insert trigger (after insert on trigger) , είναι ένα trigger το οποίο προκαλείται μετά από ένα γεγονός insert που συμβαίνει στον πίνακα contract. Στην περίπτωση μας, θέτουμε το trigger να γίνει μετά από ένα γεγονός insert στον πίνακα contract. Η λειτουργία του είναι να ενημερώσει τη στήλη con_sum του πίνακα customer προσθέτοντας το καινούργιο κόστος (στον πελάτη με το συγκεκριμένο ΑΦΜ) με το οποίο ενημερώνεται ο πίνακας contract.
- ✓ Το contract_delete trigger (after delete on trigger), είναι ένα trigger το οποίο προκαλείται μετά από ένα γεγονός delete που συμβαίνει στον πίνακα contract. Στην περίπτωση μας,

Θέτουμε το trigger να γίνει μετά από ένα γεγονός delete στον πίνακα contract. Η λειτουργία του είναι να ενημερώσει τη στήλη con_sum του πίνακα customer αφαιρώντας το καινούργιο κόστος (στον πελάτη με το συγκεκριμένο ΑΦΜ) με το οποίο ενημερώνεται ο πίνακας contract.

- ✓ Το contract_update trigger (after update on trigger), είναι ένα trigger το οποίο προκαλείται μετά από ένα γεγονός update που συμβαίνει στον πίνακα contract. Στην περίπτωση μας, θέτουμε το trigger να γίνει μετά από ένα γεγονός update στον πίνακα contract. Η λειτουργία του είναι να ενημερώσει τη στήλη con_sum του πίνακα customer εκτελώντας ένα select το οποίο υπολογίζει το άθροισμα των cost του πίνακα contract για τα συμβόλαια που έχει υπογράψει ένας συγκεκριμένος πελάτης.

Αποτελέσματα - Παραδείγματα

- ✓ Για το contract_insert trigger (after insert on trigger) παρατηρούμε τα εξής αποτελέσματα:

```
INSERT INTO contract VALUES (539, 500, '2024-12-15', '2025-12-15',  
178663326, 10);
```

```
mysql> select * from customer;
```

AFM	name	surname	address	DOY	phone	con_sum
178663326	John	Velasco	Athens	C	+302115055293	500
1023452569	Stefan	Hunnam	California	A	+602103625956	11000
1024782463	Shamkant	Navathe	Michigan	D	+852103659569	3300
1124834578	Periklis	Megalos	Volos	A	+302115858645	400
1256452238	Alexandra	Denman	Patras	A	+302100585464	4800
1415541574	Tatiana	Raptis	Thessaloniki	Z	+302105448542	1200
1445477457	Ramez	Elmasri	Dublin	C	+662108544615	900
1445525690	Henry	Cavill	Chicago	B	+902104135956	1800
1445547451	Margarita	Koumpouri	Athens	B	+302115475668	2000
1486542484	Liliana	Beckham	Chicago	C	+902138524656	1200
1542147114	Paola	Velasco	Athens	A	+302114522704	1200
1592227645	Thea	Wade	Athens	B	+302118548646	12200
1745242426	Andreas	Dimitrios	Volos	I	+302108564524	900
1784585254	Kostas	Pappas	Lamia	B	+302135874572	900
1785527877	James	Smith	Thessaloniki	D	+302103765556	0
1785592569	Orlando	Bloom	Athens	C	+302103681956	4900
1841204754	Maria	Papadopoulou	Patras	A	+302117941554	900
1884560847	Jason	Steven	Kavala	E	+302108741262	7300
1946923008	Henderson	Gibb	California	C	+602126799446	1500

19 rows in set (0.00 sec)

- ✓ Για το contract_delete trigger (after delete on trigger) παρατηρούμε τα εξής αποτελέσματα:

```
DELETE FROM contract WHERE contract_code = 528;
```

```
mysql> select * from customer;
```

AFM	name	surname	address	DOY	phone	con_sum
178663326	John	Velasco	Athens	C	+302115055293	500
1023452569	Stefan	Hunnam	California	A	+602103625956	11000
1024782463	Shamkant	Navathe	Michigan	D	+852103659569	3300
1124834578	Periklis	Megalos	Volos	A	+302115858645	400
1256452238	Alexandra	Denman	Patras	A	+302100585464	4800
1415541574	Tatiana	Raptis	Thessaloniki	Z	+302105448542	0
1445477457	Ramez	Elmasri	Dublin	C	+662108544615	900
1445525690	Henry	Cavill	Chicago	B	+902104135956	1800
1445547451	Margarita	Koumpouri	Athens	B	+302115475668	2000
1486542484	Liliana	Beckham	Chicago	C	+902138524656	1200
1542147114	Paola	Velasco	Athens	A	+302114522704	1200
1592227645	Thea	Wade	Athens	B	+302118548646	12200
1745242426	Andreas	Dimitrios	Volos	I	+302108564524	900
1784585254	Kostas	Pappas	Lamia	B	+302135874572	900
1785527877	James	Smith	Thessaloniki	D	+302103765556	0
1785592569	Orlando	Bloom	Athens	C	+302103681956	4900
1841204754	Maria	Papadopoulou	Patras	A	+302117941554	900
1884560847	Jason	Steven	Kavala	E	+302108741262	7300
1946923008	Henderson	Gibb	California	C	+602126799446	1500

```
19 rows in set (0.00 sec)
```

Και ο πίνακας contract θα έχει:

```
mysql> select * from contract;
```

contract_code	cost	start_date	end_date	AFM	insurance_code
512	11000	2000-01-01	2022-01-01	1023452569	10
513	1800	2006-01-01	2009-01-01	1445525690	20
514	4000	2004-01-01	2006-01-01	1785592569	30
515	900	2004-01-01	2010-01-01	1785592569	10
516	1000	2005-01-01	2010-01-01	1884560847	10
517	2500	2005-03-05	2014-03-05	1884560847	20
518	1800	2008-03-06	2020-03-06	1884560847	30
519	2000	2011-06-04	2017-06-04	1884560847	40
520	1200	2011-06-04	2015-06-04	1542147114	40
521	2000	2011-11-09	2014-11-09	1445547451	10
522	900	2011-07-05	2015-07-05	1445477457	30
523	1500	2012-03-08	2015-03-08	1024782463	10
524	1800	2012-08-09	2019-08-09	1024782463	20
525	900	2013-09-05	2019-09-05	1841204754	30
526	900	2013-09-05	2018-09-05	1784585254	30
527	900	2015-11-10	2022-11-10	1745242426	30
529	400	2017-04-30	2023-04-30	1124834578	40
530	1200	2017-06-02	2020-06-02	1486542484	40
531	1500	2017-06-03	2025-06-03	1946923008	10
532	4800	2017-06-04	2026-06-04	1256452238	20
533	4000	2018-09-05	2026-09-05	1592227645	10
534	4800	2018-09-05	2026-09-05	1592227645	20
535	1800	2018-09-05	2024-09-05	1592227645	30
536	1600	2018-09-05	2026-09-05	1592227645	40
539	500	2024-12-15	2025-12-15	178663326	10

```
25 rows in set (0.00 sec)
```

- ✓ Για το contract_update trigger (after update on trigger) παρατηρούμε τα εξής αποτελέσματα:

```
UPDATE contract SET cost = 623 WHERE contract_code = 517;
```

```
mysql> select * from customer;
```

AFM	name	surname	address	DOY	phone	con_sum
178663326	John	Velasco	Athens	C	+302115055293	500
1023452569	Stefan	Hunnam	California	A	+602103625956	11000
1024782463	Shamkant	Navathe	Michigan	D	+852103659569	3300
1124834578	Periklis	Megalos	Volos	A	+302115858645	400
1256452238	Alexandra	Denman	Patras	A	+302100585464	4800
1415541574	Tatiana	Raptis	Thessaloniki	Z	+302105448542	0
1445477457	Ramez	Elmasri	Dublin	C	+662108544615	900
1445525690	Henry	Cavill	Chicago	B	+902104135956	1800
1445547451	Margarita	Koumpouri	Athens	B	+302115475668	2000
1486542484	Liliana	Beckham	Chicago	C	+902138524656	1200
1542147114	Paola	Velasco	Athens	A	+302114522704	1200
1592227645	Thea	Wade	Athens	B	+302118548646	12200
1745242426	Andreas	Dimitrios	Volos	I	+302108564524	900
1784585254	Kostas	Pappas	Lamia	B	+302135874572	900
1785527877	James	Smith	Thessaloniki	D	+302103765556	0
1785592569	Orlando	Bloom	Athens	C	+302103681956	4900
1841204754	Maria	Papadopoulou	Patras	A	+302117941554	900
1884560847	Jason	Steven	Kavala	E	+302108741262	5423
1946923008	Henderson	Gibb	California	C	+602126799446	1500

19 rows in set (0.00 sec)

Στον πίνακα contract παρατηρούμε τα εξής αποτελέσματα:

```
mysql> select * from contract;
```

contract_code	cost	start_date	end_date	AFM	insurance_code
512	11000	2000-01-01	2022-01-01	1023452569	10
513	1800	2006-01-01	2009-01-01	1445525690	20
514	4000	2004-01-01	2006-01-01	1785592569	30
515	900	2004-01-01	2010-01-01	1785592569	10
516	1000	2005-01-01	2010-01-01	1884560847	10
517	623	2005-03-05	2014-03-05	1884560847	20
518	1800	2008-03-06	2020-03-06	1884560847	30
519	2000	2011-06-04	2017-06-04	1884560847	40
520	1200	2011-06-04	2015-06-04	1542147114	40
521	2000	2011-11-09	2014-11-09	1445547451	10
522	900	2011-07-05	2015-07-05	1445477457	30
523	1500	2012-03-08	2015-03-08	1024782463	10
524	1800	2012-08-09	2019-08-09	1024782463	20
525	900	2013-09-05	2019-09-05	1841204754	30
526	900	2013-09-05	2018-09-05	1784585254	30
527	900	2015-11-10	2022-11-10	1745242426	30
529	400	2017-04-30	2023-04-30	1124834578	40
530	1200	2017-06-02	2020-06-02	1486542484	40
531	1500	2017-06-03	2025-06-03	1946923008	10
532	4800	2017-06-04	2026-06-04	1256452238	20
533	4000	2018-09-05	2026-09-05	1592227645	10
534	4800	2018-09-05	2026-09-05	1592227645	20
535	1800	2018-09-05	2024-09-05	1592227645	30
536	1600	2018-09-05	2026-09-05	1592227645	40
539	500	2024-12-15	2025-12-15	178663326	10

25 rows in set (0.00 sec)

2. Functions

Βήμα 7

Δημιουργήστε μια *function* η οποία θα επιστρέφει τη διάρκεια των συμβολαίων λαμβάνοντας υπόψιν την ημερομηνία έναρξης και λήξης του συμβολαίου (π.χ. 6 μήνες).

Εντολές

```
#function: date_check that counts years duration

DROP FUNCTION IF EXISTS date_check;
delimiter //
CREATE FUNCTION date_check()
RETURNS varchar(255)
DETERMINISTIC
BEGIN

DECLARE record_not_found int default 0;
DECLARE dStart Date;
DECLARE dEND Date;
DECLARE contractDuration varchar(255) default ' ';
DECLARE my_cursor cursor for SELECT start_date, END_date FROM contract;
DECLARE continue handler for not found SET record_not_found = 1;

OPEN my_cursor;
allclients: LOOP
    FETCH my_cursor INTO dStart, dEND;
    SELECT timestampdiff(YEAR,dStart, dEND) INTO @f;
    IF record_not_found THEN leave allclients;
    END IF;
    SET contractDuration = CONCAT(contractDuration, @f, ", ");
END LOOP allclients;
CLOSE my_cursor;

return substr(contractDuration,1,255);
END
//

delimiter ;

SELECT date_check();
```

Τεκμηρίωση

Μια *function* λαμβάνει ένα σύνολο παραμέτρων ή μη, ενώ μπορεί να επιστρέψει μόνο μια τιμή.

Στη συγκεκριμένη, η *function* με όνομα `date_check` δε λαμβάνει κάποια είσοδο., ωστόσο δηλώνονται σε αυτή οι μεταβλητές

- `dStart Date` : μεταβλητή που θα παίρνει τιμές του `start_date`.

- **dEnd Date** : μεταβλητή που θα παίρνει τιμές του end_date.
- **contractDuration** : η «λίστα» στην οποία θα αποθηκευτούν τα αποτελέσματα και θα επιστρέψει η function.

Αρχικά ορίζουμε έναν *cursor* , ο οποίος μας επιτρέπει να διατρέξουμε ένα σύνολο από σειρές που επιστρέφονται από ένα *query* και κατά συνέπεια, να επεξεργαστούμε κάθε σειρά ξεχωριστά.

Με τον τρόπο αυτό, οι μεταβλητές *dStart* και *dEnd* που ορίσαμε πριν θα πάρουν τις εγγραφές του *select* που εκτελέσαμε στη δήλωση του *cursor*, γεγονός το οποίο θα μας επιτρέψει να επεξεργαστούμε τις τιμές που μας επιστρέφονται.

Στη συνέχεια δημιουργούμε ένα βρόχο στον οποίο θα τρέχει για κάθε entry του πίνακα *contract* και θα υπολογίζεται η διάρκεια του συμβολαίου. Η τιμή αυτή αποθηκεύεται στην *@f*.

Τέλος σε κάθε επανάληψη γίνεται η συνένωση της τιμής της προσωρινής *contractDuration* με την τιμή της *@f*.

Αποτελέσματα - Παραδείγματα

Ο πίνακας *contract* έχει :

```
mysql> select * from Contract;
```

contract_code	cost	start_date	end_date	AFM	insurance_code
512	11000	2000-01-01	2022-01-01	1023452569	10
513	1800	2006-01-01	2009-01-01	1445525690	20
514	4000	2004-01-01	2006-01-01	1785592569	30
515	900	2004-01-01	2010-01-01	1785592569	10
516	1000	2005-01-01	2010-01-01	1884560847	10
517	2500	2005-03-05	2014-03-05	1884560847	20
518	1800	2008-03-06	2020-03-06	1884560847	30
519	2000	2011-06-04	2017-06-04	1884560847	40
520	1200	2011-06-04	2015-06-04	1542147114	40
521	2000	2011-11-09	2014-11-09	1445547451	10
522	900	2011-07-05	2015-07-05	1445477457	30
523	1500	2012-03-08	2015-03-08	1024782463	10
524	1800	2012-08-09	2019-08-09	1024782463	20
525	900	2013-09-05	2019-09-05	1841204754	30
526	900	2013-09-05	2018-09-05	1784585254	30
527	900	2015-11-10	2022-11-10	1745242426	30
528	1200	2016-12-10	2019-12-10	1415541574	40
529	400	2017-04-30	2023-04-30	1124834578	40
530	1200	2017-06-02	2020-06-02	1486542484	40
531	1500	2017-06-03	2025-06-03	1946923008	10
532	4800	2017-06-04	2026-06-04	1256452238	20
533	4000	2018-09-05	2026-09-05	1592227645	10
534	4800	2018-09-05	2026-09-05	1592227645	20
535	1800	2018-09-05	2024-09-05	1592227645	30
536	1600	2018-09-05	2026-09-05	1592227645	40

25 rows in set (0.01 sec)

```
select date_check();
```

```
mysql> select date_check();
```

date_check()
22, 3, 2, 6, 5, 9, 12, 6, 4, 3, 4, 3, 7, 6, 5, 7, 3, 6, 3, 8, 9, 8, 8, 6, 8,

1 row in set (0.00 sec)

3. Procedures

Βήμα 8

Δημιουργήστε μια *procedure* η οποία θα δέχεται σαν είσοδο το ΑΦΜ ενός πελάτη και μια ημερομηνία και θα επιστρέφει τον αριθμό των συμβολαίων που κατέχει εκείνο το μήνα, καθώς επίσης και το συνολικό ποσό που πρέπει να πληρώσει το συγκεκριμένο μήνα.

Εντολές

```
#table that contains the results of each getPayment_proc call
DROP TABLE IF EXISTS monthlyPayment;
CREATE table monthlyPayment
(
    m_afm INT,
    m_date DATE,
    m_no INT,
    m_cost decimal(7,2),
    PRIMARY KEY (m_afm, m_date)
);

#creating procedure getPayment_proc
DROP PROCEDURE IF EXISTS getPayment_proc;
delimiter !
CREATE procedure getPayment_proc(
    IN p_afm INT,
    IN p_date DATE,
    OUT p_no INT,
    OUT p_cost decimal(7,2)
)
BEGIN

    #number of active contracts
    SET p_no = (SELECT count(*) FROM insurance INNER JOIN contract
    ON insurance.insurance_code = contract.insurance_code
    WHERE AFM = p_afm
    AND p_date BETWEEN start_date AND END_date
    );

    #total cost of certain month
    SET p_cost = (SELECT sum(annual_cost/12) FROM insurance INNER JOIN
contract
    ON insurance.insurance_code = contract.insurance_code
    WHERE AFM = p_afm
    AND p_date BETWEEN start_date AND END_date
    );

    INSERT INTO monthlyPayment VALUES (p_afm, p_date, p_no, p_cost);
END !
delimiter ;
```



```

SELECT * FROM monthlyPayment;
CALL getPayment_proc (1592227645,'2025-09-27', @out_no, @out_cost);
SELECT * FROM monthlyPayment;

#duplicate entry
CALL getPayment_proc (1592227645,'2025-09-27', @out_no, @out_cost);

CALL getPayment_proc (1592227645,'2023-09-27', @out_no, @out_cost);
SELECT * FROM monthlyPayment;

CALL getPayment_proc(1445547451, '2013-12-27',@out_no, @out_cost);
SELECT * FROM monthlyPayment;

```

Τεκμηρίωση

Δημιουργούμε έναν πίνακα «monthlyPayment» ο οποίος θα περιέχει τα αποτελέσματα κάθε κλήσης της συνάρτησης `getPayment_proc`.

Η συνάρτηση «`getPayment_proc`» έχει ως είσοδο το ΑΦΜ του πελάτη και μια ημερομηνία, ενώ ως έξοδο έχει τον αριθμό των συμβολαίων που είναι ενεργά καθώς και το ποσό που χρήζει να πληρώσει ο πελάτης εκείνο τον μήνα. Οι τιμές των εξόδων αποθηκεύονται στις μεταβλητές του προγραμματιστή στις `@out_no` και `@out_cost` αντίστοιχα κατά την κλήση της συνάρτησης.

Αποτελέσματα - Παραδείγματα

```
select * from monthlyPayment;
```

```

mysql> select * from monthlyPayment;
Empty set (0.00 sec)

mysql>

```

```

call getPayment_proc (1592227645,'2025-09-27', @out_no, @out_cost);
select * from monthlyPayment;

```

```

mysql> call getPayment_proc (1592227645,'2025-09-27', @out_no, @out_cost);
Query OK, 1 row affected (0.03 sec)

mysql> select * from monthlyPayment;
+-----+-----+-----+-----+
| m_afm   | m_date   | m_no | m_cost |
+-----+-----+-----+-----+
| 1592227645 | 2025-09-27 | 3 | 108.33 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

```
#duplicate entry -error
call getPayment_proc (1592227645,'2025-09-27', @out_no, @out_cost);
call getPayment_proc (1592227645,'2023-09-27', @out_no, @out_cost);
select * from monthlyPayment;
```

```
mysql> #duplicate entry -error
mysql> call getPayment_proc (1592227645,'2025-09-27', @out_no, @out_cost);
ERROR 1062 (23000): Duplicate entry '1592227645-2025-09-27' for key 'monthlypayment.PRIMARY'
mysql> call getPayment_proc (1592227645,'2023-09-27', @out_no, @out_cost);
Query OK, 1 row affected (0.00 sec)

mysql> select * from monthlyPayment;
+-----+-----+-----+-----+
| m_afm | m_date | m_no | m_cost |
+-----+-----+-----+-----+
| 1592227645 | 2023-09-27 | 4 | 133.33 |
| 1592227645 | 2025-09-27 | 3 | 108.33 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
call getPayment_proc(1445547451, '2013-12-27',@out_no, @out_cost);
select * from monthlyPayment;
```

```
mysql> call getPayment_proc(1445547451, '2013-12-27',@out_no, @out_cost);
Query OK, 1 row affected (0.00 sec)

mysql> select * from monthlyPayment;
+-----+-----+-----+-----+
| m_afm | m_date | m_no | m_cost |
+-----+-----+-----+-----+
| 1445547451 | 2013-12-27 | 1 | 41.67 |
| 1592227645 | 2023-09-27 | 4 | 133.33 |
| 1592227645 | 2025-09-27 | 3 | 108.33 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Οι τιμές που περιέχονται στις μεταβλητές του προγραμματιστή @out_no, @out_cost στην τελευταία κλήση της getPayment_proc.

```
mysql> select @out_cost
-> ;
+-----+
| @out_cost |
+-----+
| 41.67 |
+-----+
1 row in set (0.00 sec)

mysql> select @out_no;
+-----+
| @out_no |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)
```

ΣΥΜΠΕΡΑΣΜΑΤΑ

Έχοντας διεκπεραιώσει τη δεύτερη εργασία, μπορούμε να αντιληφθούμε καλύτερα τη δομή που χρησιμοποιήσαμε στην προηγούμενη εργασία, καθώς πλέον χρησιμοποιούμε *triggers*, *functions* και *procedures* με σκοπό την εμφάνιση και διαχείριση συγκεκριμένων πληροφοριών.

Μελετήσαμε σε βάθος τον σκοπό και τη δόμηση ενός *trigger* για τις περιπτώσεις εισαγωγής, διαγραφής και ενημέρωσης παραπάνω του ενός πίνακα.

Υλοποιήσαμε μια *function* χωρίς ορίσματα και παρατηρήσαμε τη λειτουργία της επαναληπτικά.

Τέλος, υλοποιήσαμε μια *procedure*, η οποία δέχεται ορίσματα και επιστρέφει τον αριθμό και την μηνιαία καταβολή χρημάτων για τον συγκεκριμένο μήνα που ορίζει η ημερομηνία που βρίσκεται σαν παράμετρος.