MINI-PROJECT IN DATABASES: INSURANCE MODEL

United Insurances

By Pinhas FEDIDAT, ID 7665692 and Yehuda KLIRS, ID 026618686.

With Aryeh Wiesen, Jerusalem College of Technology, 2013

פנחס פדידה 7665692, יהודה קלירס 026618686

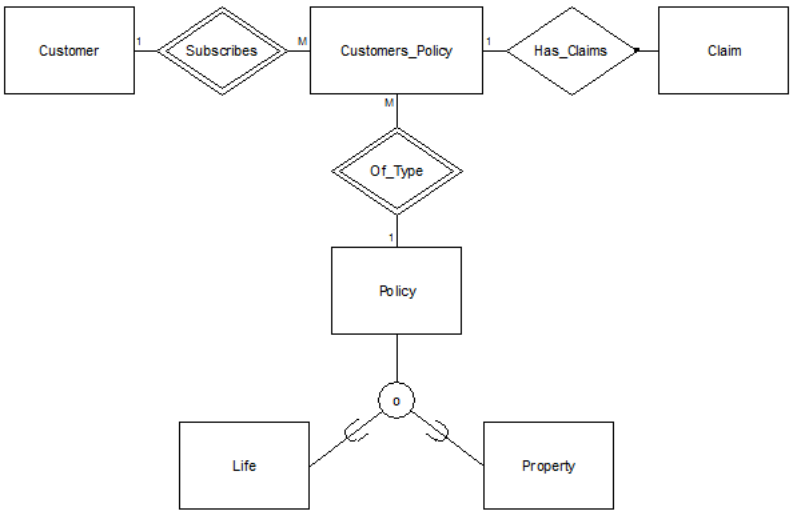
Stage 1 - Proposition of Organization – Insurance Model

The proposed subject for this mini-project is an insurance company model, organizing its customers, subscribed policies and claims. The company proposes insurance for various types of property (types of vehicles, houses or others) as well as life insurance depending on the occupation of the subscribing customer.

The database is composed of 6 tables: :

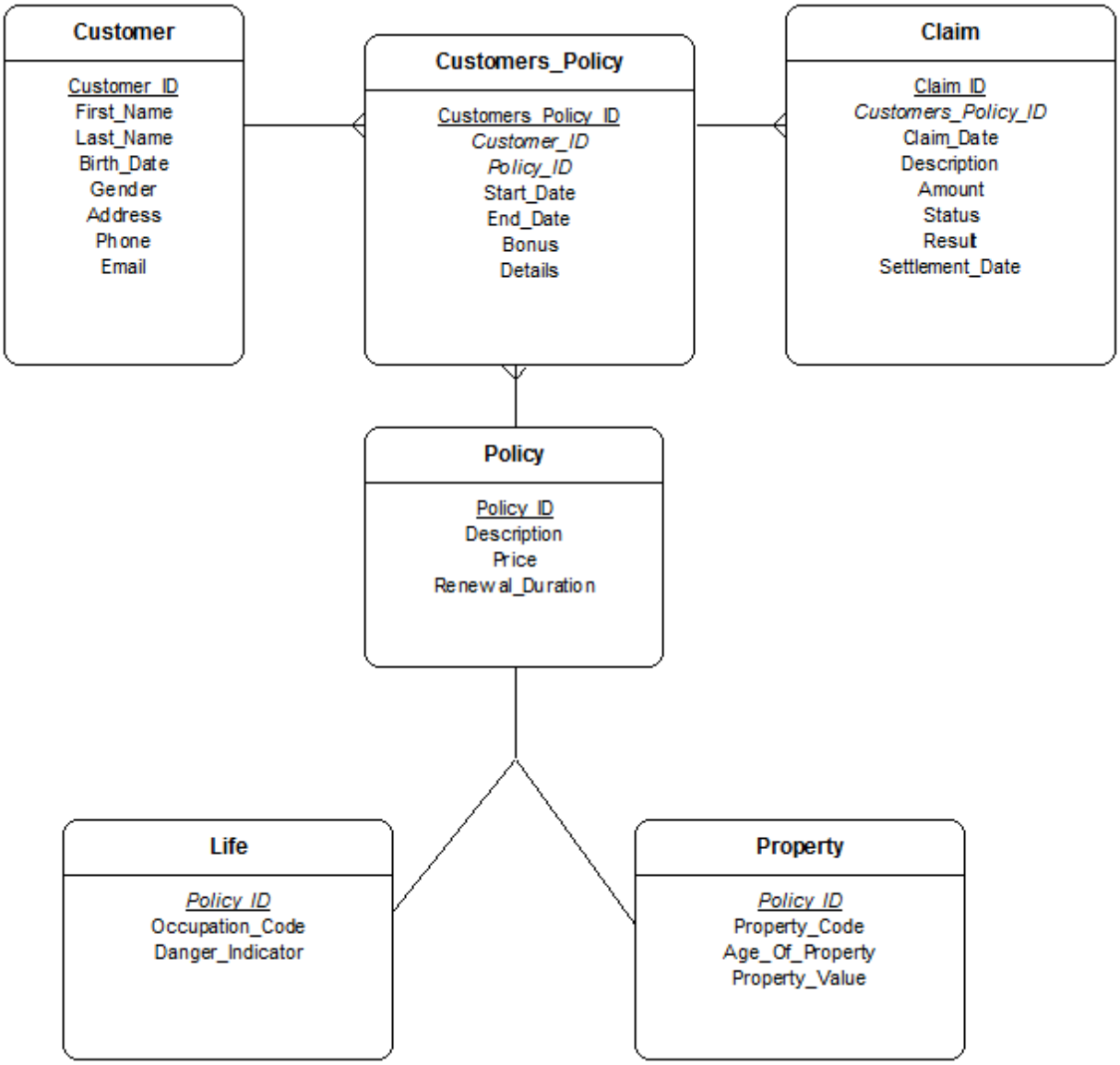
- **Customers**: ID, first name, last name, birth date, gender and contact details (address, phone, email).
- **Policies**: Policy ID, description, price and duration of the policy until renewal (in months). There are 2 specifications of this table:
  - **Life insurance policies**: occupation code and respective danger indicator.
  - **Property insurance policies**: property code, age and value of property.
- **Subscribed policies (customers_policies)**: customers_policys_ID, subscription start and end dates, bonus for long periods without claims and other details, customer and policy keys.
- **Complaints / claims**: claim_ID, customers_policys_ID, .description, requested indemnification amount, status (e.g. pending, refused, accepted), and description of the result (e.g "Customer was fully indemnified"), claim and settlement dates.
-

Entity–relationship model (ERD)



פנחס פדידה 7665692, יהודה קלירס 026618686

Data Structure Diagram (DSD)



SQL Script

```
--
-- Target: Oracle
-- Syntax: sqlplus user@tnsnames_entry/password @filename.sql
--
-- Date  : Apr 04 2013 15:07
-- Script Generated by Database Design Studio 2.21.3
--



--
-- Create Table    : 'Customer'
-- Customer_ID     :
-- First_Name      :
-- Last_Name       :
-- Birth_Date      :
-- Gender          :
-- Address         :
-- Phone           :
-- Email           :
--
CREATE TABLE Customer (
    Customer_ID    NUMBER(38) NOT NULL,
    First_Name     VARCHAR2(20) NOT NULL,
    Last_Name      VARCHAR2(20) NOT NULL,
    Birth_Date     DATE NOT NULL,
    Gender         NUMBER(1) NOT NULL,
    Address        VARCHAR2(100) NOT NULL,
    Phone          VARCHAR2(20) NOT NULL,
    Email          VARCHAR2(100) NOT NULL,
CONSTRAINT pk_Customer PRIMARY KEY (Customer_ID))
/


--
-- Create Table    : 'Policies'
-- Policy_ID       :
-- Description     :
-- Price           :
-- Renewal_Duration :
--
CREATE TABLE Policies (
    Policy_ID      NUMBER(38) NOT NULL,
    Description    CLOB NOT NULL,
    Price          NUMBER(38) NOT NULL,
    Renewal_Duration NUMBER(38) NOT NULL,
CONSTRAINT pk_Policies PRIMARY KEY (Policy_ID))
/


--
-- Create Table    : 'Customers_Policies'
```

```
-- Customers_Policys_ID :
-- Customer_ID     :  (references Customer.Customer_ID)
-- Policy_ID       :  (references Policies.Policy_ID)
-- Start_Date      :
-- End_Date        :
-- Bonus           :
-- Details         :
--
CREATE TABLE Customers_Policies (
    Customers_Policys_ID NUMBER(38) NOT NULL,
    Customer_ID    NUMBER(38) NOT NULL,
    Policy_ID      NUMBER(38) NOT NULL,
    Start_Date     DATE NOT NULL,
    End_Date       DATE NOT NULL,
    Bonus          FLOAT NOT NULL,
    Details        CLOB,
CONSTRAINT pk_Customers_Policies PRIMARY KEY (Customers_Policys_ID),
CONSTRAINT fk_Customers_Policies FOREIGN KEY (Customer_ID)
    REFERENCES Customer (Customer_ID)
    ON DELETE CASCADE,
CONSTRAINT fk_Customers_Policies2 FOREIGN KEY (Policy_ID)
    REFERENCES Policies (Policy_ID)
    ON DELETE CASCADE)
/


--
-- Create Table    : 'Life'
-- Policy ID       :  (references Policies.Policy_ID)
-- Occupation Code :
-- Danger_Indicator :
--
CREATE TABLE Life (
    Policy_ID      NUMBER(38) NOT NULL,
    Occupation_Code NUMBER(38) NOT NULL,
    Danger_Indicator FLOAT NOT NULL,
CONSTRAINT pk_Life PRIMARY KEY (Policy_ID),
CONSTRAINT fk_Life FOREIGN KEY (Policy_ID)
    REFERENCES Policies (Policy_ID))
/


--
-- Create Table    : 'Property'
-- Policy_ID       :  (references Policies.Policy_ID)
-- Property_Code   :
-- Age_Of_Property :
-- Property_Value  :
--
CREATE TABLE Property (
    Policy_ID      NUMBER(38) NOT NULL,
    Property_Code  NUMBER(1) NOT NULL,
    Age_Of_Property NUMBER(38) NOT NULL,
    Property_Value NUMBER(38) NOT NULL,
```

```
CONSTRAINT pk_Property PRIMARY KEY (Policy_ID),
CONSTRAINT fk_Property FOREIGN KEY (Policy_ID)
    REFERENCES Policies (Policy_ID))
/


--
-- Create Table    : 'Claims'
-- Claim_ID        :
-- Customers_Policys_ID :  (references
Customers Policies.Customers_Policys_ID)
-- Claim_Date      :
-- Description     :
-- Amount          :
-- Status          :
-- Result          :
-- Settlement_Date :
--
CREATE TABLE Claims (
    Claim_ID       NUMBER(38) NOT NULL,
    Customers_Policys_ID NUMBER(38) NOT NULL,
    Claim_Date     DATE NOT NULL,
    Description    CLOB NOT NULL,
    Amount         NUMBER(38) NOT NULL,
    Status         NUMBER(1),
    Result         CLOB,
    Settlement_Date DATE,
CONSTRAINT pk_Claims PRIMARY KEY (Claim_ID),
CONSTRAINT fk_Claims FOREIGN KEY (Customers_Policys_ID)
    REFERENCES Customers_Policies (Customers_Policys_ID)
    ON DELETE CASCADE)
/


--
-- Permissions for: 'public'
--
GRANT ALL ON Customer TO public
/
GRANT ALL ON Policies TO public
/
GRANT ALL ON Customers_Policies TO public
/
GRANT ALL ON Life TO public
/
GRANT ALL ON Property TO public
/
GRANT ALL ON Claims TO public
/

exit;
```

Queries

Customers

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	BIRTH_DATE	GENDER	ADDRESS	PHONE	EM
1	287333226	Claire	Stills	26/01/1931	1	17 Blaine Street 27646 Roma Italy	(+345) 77-748-8257	csti
2	209875394	Claude	Bening	18/05/1969	0	22 Kim Road 6N4 8Y7 Thames Ditton United Kingdom	(+531) 68-140-8231	cla
3	245018189	Ashley	Martin	10/06/1944	0	5 Warley Street 1088CZ Eindhoven Netherlands	(+752) 12-441-7628	ash
4	213747027	Cary	Carrere	06/08/1936	1	108 McCormack Ave 10295 Charlotte USA	(+777) 72-877-2343	car
5	255240677	Natalie	Guinness	07/03/1958	0	570 Francis Drive 30798 Karlsruhe Germany	(+120) 03-752-5281	nat
6	178804678	Cliff	Swayze	16/05/1955	1	567 Louise Road 33648 Stony Point USA	(+613) 30-169-3987	csw
7	268322605	Mary Beth	Francis	26/01/1932	0	68 Boone Road 5W3 6H0 Winnipeg Canada	(+226) 70-724-8733	ma
8	116422180	Mekhi	Rock	01/05/1944	0	98 Paris Street 33092 Schenectady USA	(+642) 78-526-6213	me
9	201911690	Ethan	Depp	30/07/1931	1	27 Carr Blvd 5460 Reno USA	(+425) 32-893-9832	eth
10	168392377	Teena	Peebles	01/04/1954	0	90 Adina Road 37438 Lyon France	(+587) 37-386-3622	tee
11	153270408	Lance	Sevigny	07/03/1984	0	786 Chennai Blvd 8230 Friedrichshafe Germany	(+629) 52-824-2632	lan
12	277654377	Adam	Ribisi	08/01/1974	0	20 Forster Blvd 39886 Soroe Denmark	(+018) 22-976-5480	ada
13	125371369	Kiefer	Walken	29/06/1938	1	712 Getty Ave 18692 Rimini Italy	(+896) 02-423-3124	kwa
14	236214649	Adam	Stermare	05/01/1942	0	45 Romar Blvd 24073 Des Plaines USA	(+908) 26-312-9437	ada

Policies

	POLICY_ID	DESCRIPTION	PRICE	RENEWAL_DURATION
1	1	<CLOB>	41	1
2	2	<CLOB>	56	16
3	3	<CLOB>	43	30
4	4	<CLOB>	42	10
5	5	<CLOB>	31	22
6	6	<CLOB>	54	26
7	7	<CLOB>	36	3
8	8	<CLOB>	33	18
9	9	<CLOB>	31	7
10	10	<CLOB>	42	22
11	11	<CLOB>	55	26
12	12	<CLOB>	40	5

Property

	POLICY_ID	PROPERTY_CODE	AGE_OF_PROPERTY	PROPERTY_VALUE
1	1	1	17	814797
2	2	2	20	930894
3	3	5	15	937486
4	4	1	18	372006
5	5	1	18	608206
6	6	5	19	84493
7	7	4	6	180176
8	8	2	24	522332
9	9	2	19	831504
10	10	3	3	222657

Life

	POLICY_ID	OCCUPATION_CODE	DANGER_INDICATOR
1	19	38	7.72
2	20	24	0.42
3	21	25	0.5
4	22	43	1.72
5	23	18	7.62
6	24	22	5.34
7	25	32	6.84
8	26	34	8.85
9	27	21	2.37
10	28	45	1.71
11	29	29	5.9

Customers_Policies

	START_DATE	CUSTOMER_ID	POLICY_ID	END_DATE	BONUS	DETAILS	CUSTOMERS_POLICYS_ID
1	28/12/1949	222866171	23	05/09/2003	5.18	<CLOB>	1
2	23/09/1967	205316226	21	20/11/1983	1.11	<CLOB>	2
3	28/09/1979	244513477	25	15/12/2009	6.53	<CLOB>	3
4	01/11/1953	144779383	28	13/06/2006	5.57	<CLOB>	4
5	29/08/1965	148700777	11	29/11/2007	6.95	<CLOB>	5
6	23/07/1956	127570839	12	29/01/1988	4.59	<CLOB>	6
7	26/04/1987	148700777	16	03/09/1998	6.26	<CLOB>	7
8	04/03/1938	249694516	3	14/07/2007	9.38	<CLOB>	8
9	22/09/1979	125297113	1	21/06/2004	6.34	<CLOB>	9
10	18/09/1974	317077240	30	07/05/2002	2.31	<CLOB>	10
11	04/06/1933	293239263	6	12/11/1992	3.6	<CLOB>	11
12	13/01/1936	317471701	2	07/05/2000	1.76	<CLOB>	12

Claims

	CLAIM_ID	CLAIM_DATE	DESCRIPTION	AMOUNT	STATUS	RESULT	SETTLEMENT_DATE	CUSTOMERS_POLICYS_ID
1	1	23/02/2005	<CLOB>	43392	2	<CLOB>	20/09/2008	896
2	2	12/11/2007	<CLOB>	62176	1	<CLOB>	12/07/1988	199
3	3	23/07/1984	<CLOB>	95303	6	<CLOB>	27/09/2001	722
4	4	08/12/1998	<CLOB>	68249	0	<CLOB>	20/03/2002	668
5	5	27/11/1980	<CLOB>	34485	1	<CLOB>	27/03/1999	618
6	6	02/07/2002	<CLOB>	50080	5	<CLOB>	03/07/1989	823
7	7	10/10/2006	<CLOB>	30426	8	<CLOB>	31/07/2002	693
8	8	02/08/2008	<CLOB>	74427	0	<CLOB>	19/12/2000	324
9	9	08/12/2007	<CLOB>	92545	4	<CLOB>	19/04/1993	667
10	10	03/03/2000	<CLOB>	35520	6	<CLOB>	05/08/1996	792
11	11	13/03/1999	<CLOB>	60121	8	<CLOB>	29/04/2002	155

Level 2 – Queries, indexes, changes, constraints, views

1. Eight queries that demonstrate practical use of the insurance database

1. Join, order by: allows us to see all the policies subscribed by customers, together with customer information.

```
select customer_id, policy_id, start_date
from customer natural join customers_policy
order by start_date asc;
```

2. Join, aggregate, group by: allows us to see the number of policies that customers subscribed to for each type of important policy (which have over 30 subscribers).

```
select policy.policy_id, count(*)
from policy left join customers_policy
on policy.policy_id = customers_policy.policy_id
group by policy.policy_id
having count(*) > 30;
```

3. Sub-query, aggregate: shows the sum of claims for policies subscribed after 01/01/2000.

```
select sum(amount) from (
       select amount, status
       from claim
       where customers_policy_id in(
             select customers_policy_id from customers_policy
             where start_date > DATE '2000-01-01'
       )
);
```

4. Where, order by: shows solved claims (where `status` is `0`), sorted by submit date from oldest to most recent.

```
select * from claim
where status = 0
order by claim_date asc;
```

5. Where, sub-query: selects the name and emails of customers with the top ten bonuses.

```
select First_Name, Last_Name, email from Customer
where Customer_ID in (
    select Customer_ID from (
           select * from Customers_Policy
           order by bonus desc)
    where rownum <= 10);
```

6. Group by, aggregate: gives us a list of bad customers (of which the average of their bonuses is below 5).

```
select Customer_ID, avg(bonus) from Customers_Policy
group by Customer_ID
having avg(bonus) < 5;
```

7. Aggregate, group by: shows the count of customers for each gender.

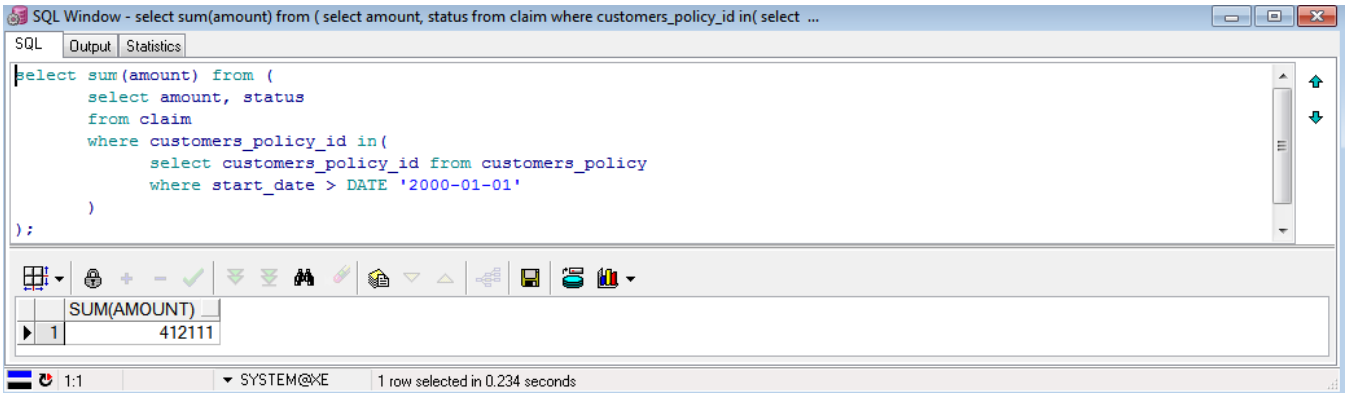
```
select gender, count(*) from Customer
group by gender;
```

8. Aggregate, group by: shows the total price to charge for each customer.

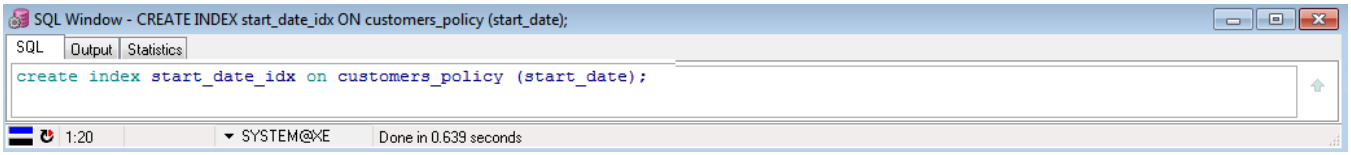
```
select Customer_ID, sum(Policy.price) from Customers_Policy natural join
Policy
group by Customer_ID;
```

2. Index and speed improvement

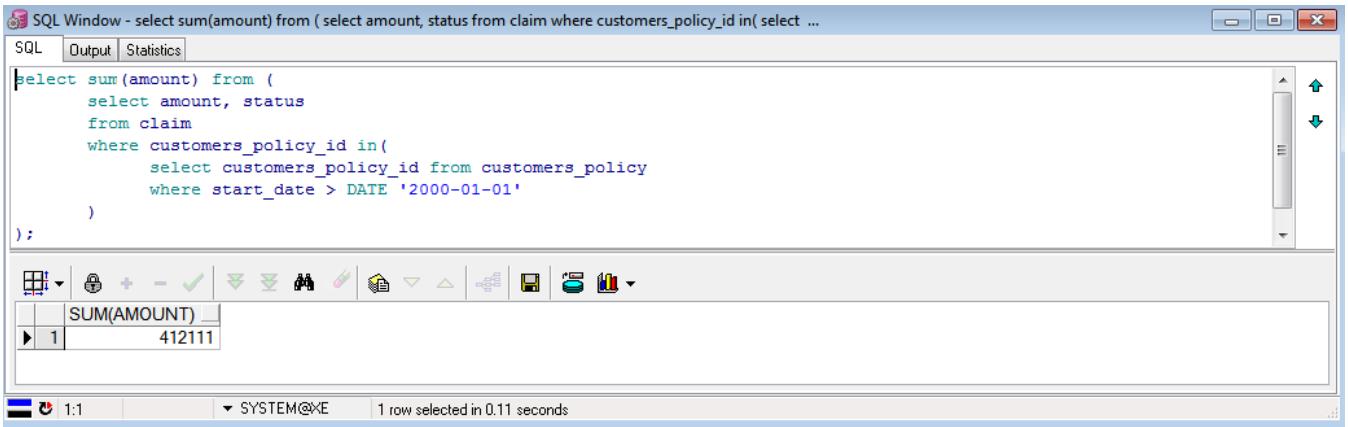
Here the query number 3 above is executed:



Then the Oracle server is stopped and started (`cmd>net start/stop OracleServiceXE`) in order to clear the buffers, which prevents unwanted cache optimization from the previous querying and we create an index over the `customers_policy.start_date`, which will optimize the where statement in the aforementioned query:



Finally, we run the query again. Notice the expected performance gain of 134ms (57%):



3. Two Update and two Delete queries

1. Update customer's email address:

```
update customer
set email = 'dacruz@anomel.com'
where customer_id = 237510106;
```

2. Update claim `status`:

```
update claim
set status = 0
where customers_policy_id = 153;
```

3. Delete claims related to customer's policy 123:

```
delete claim
where customers_policy_id = 123;
```

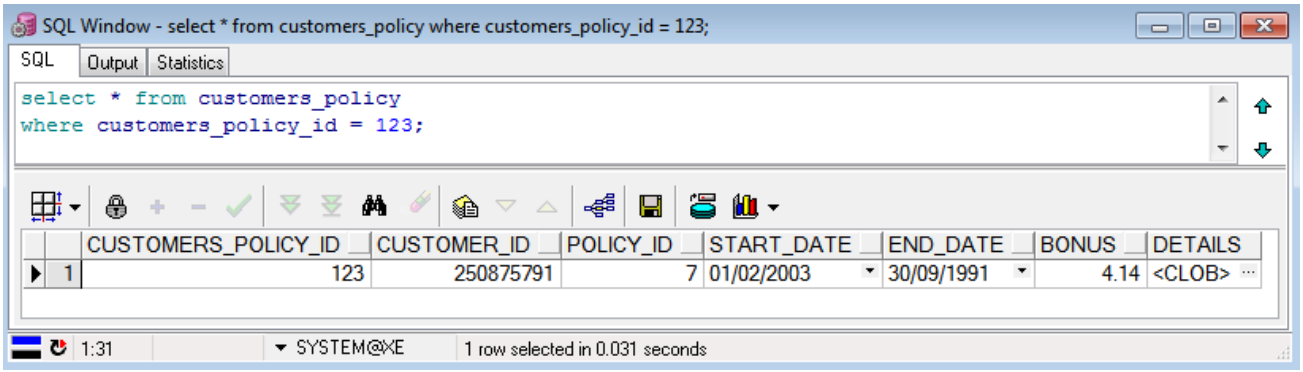
4. And then delete the customer's policy itself:

```
delete customers_policy
where customers_policy_id = 123;
```

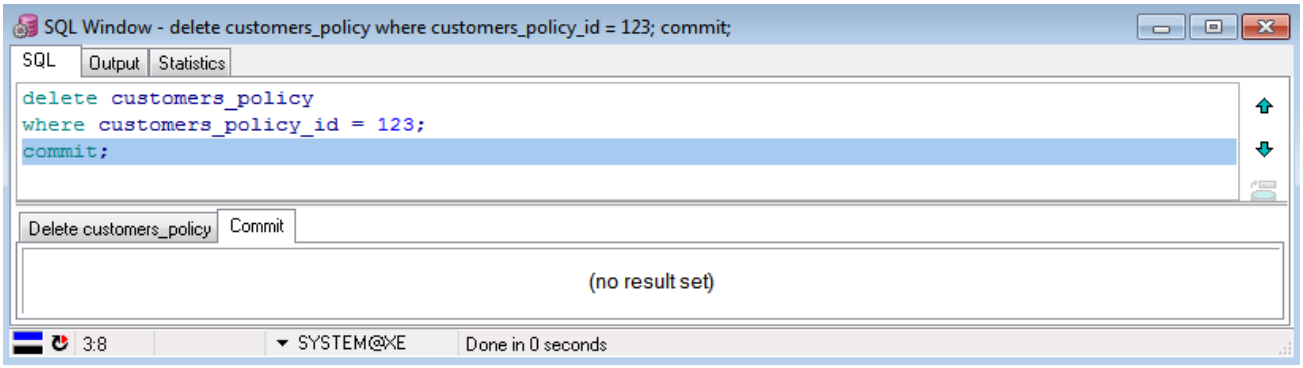
4. Commit and rollback

1. Commit

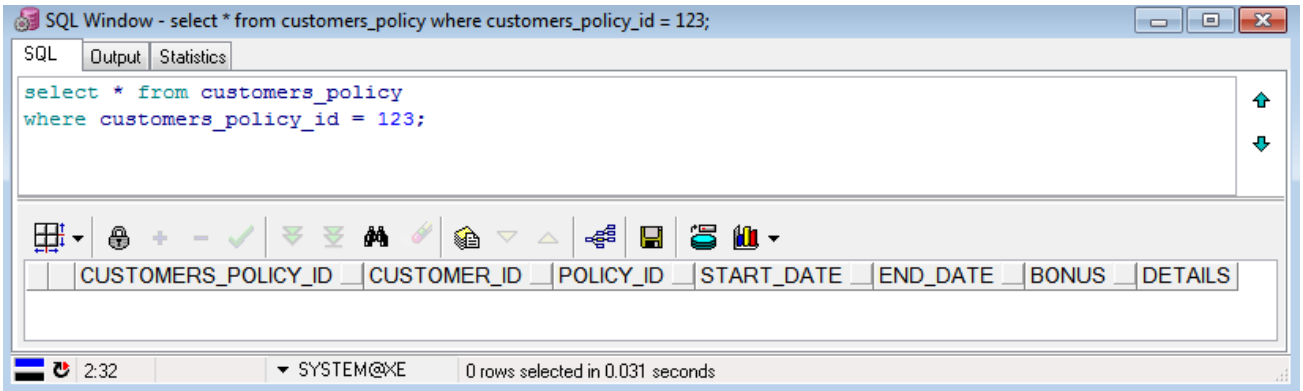
The customer's policy record for ID 123 looks this way *before and after deletion*, as long as we don't commit:



Then we execute the delete query and commit command:



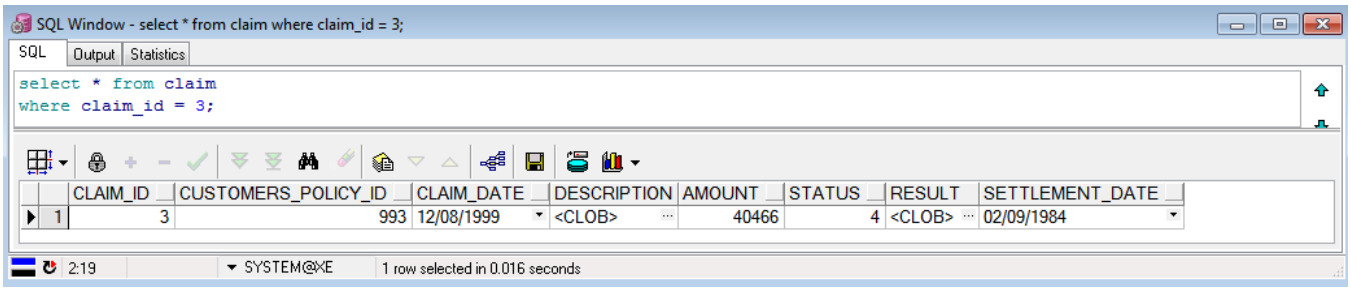
And this is what we get after commit:



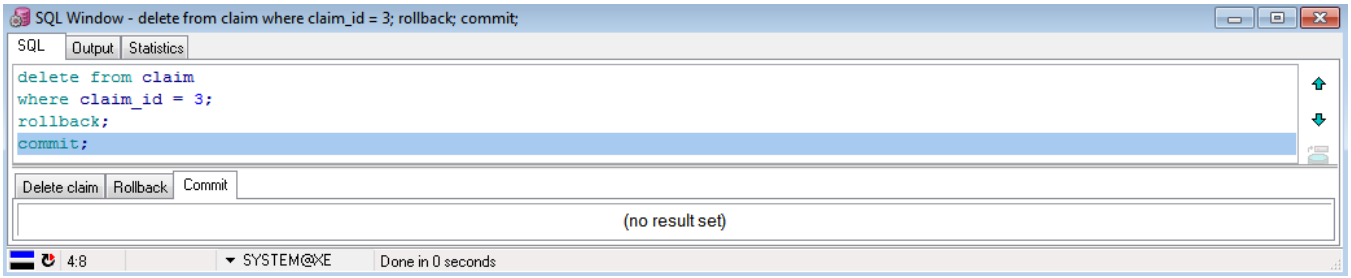
Of course, we can see that the customer's policy record is now gone.

2. Rollback

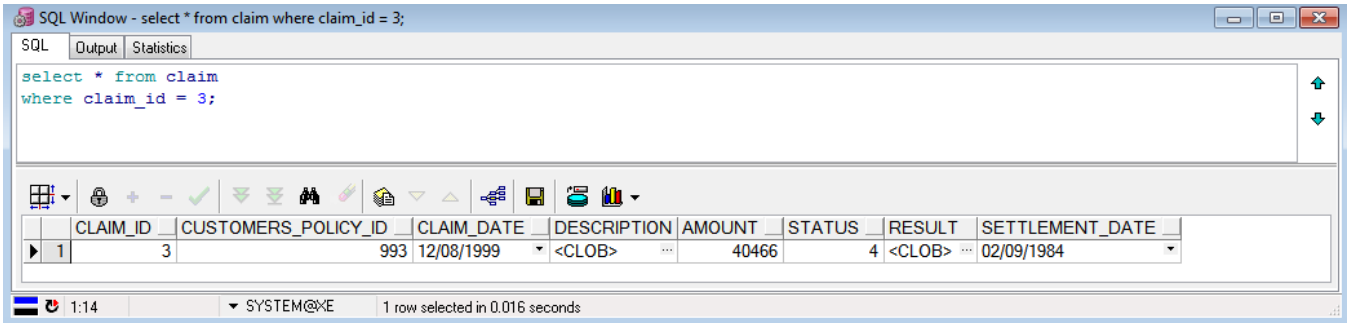
This is what we get before deletion:



Then we delete, rollback and commit (in order to show that even if we commit, as long as we rolled back, the changes before the rollback are cancelled):



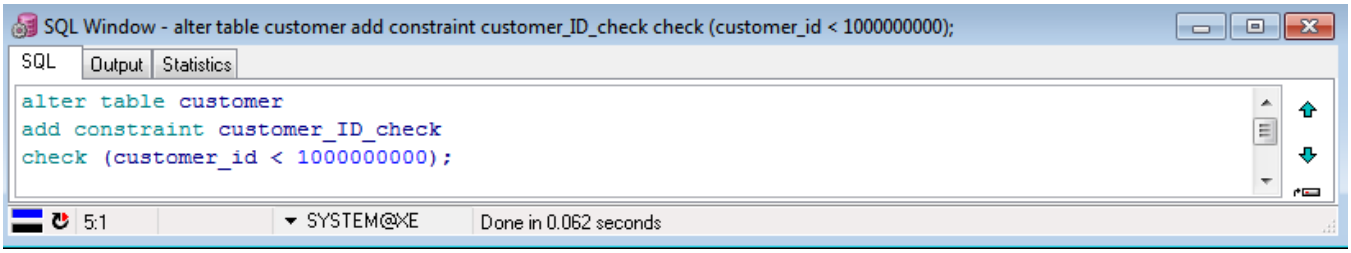
And finally query again:



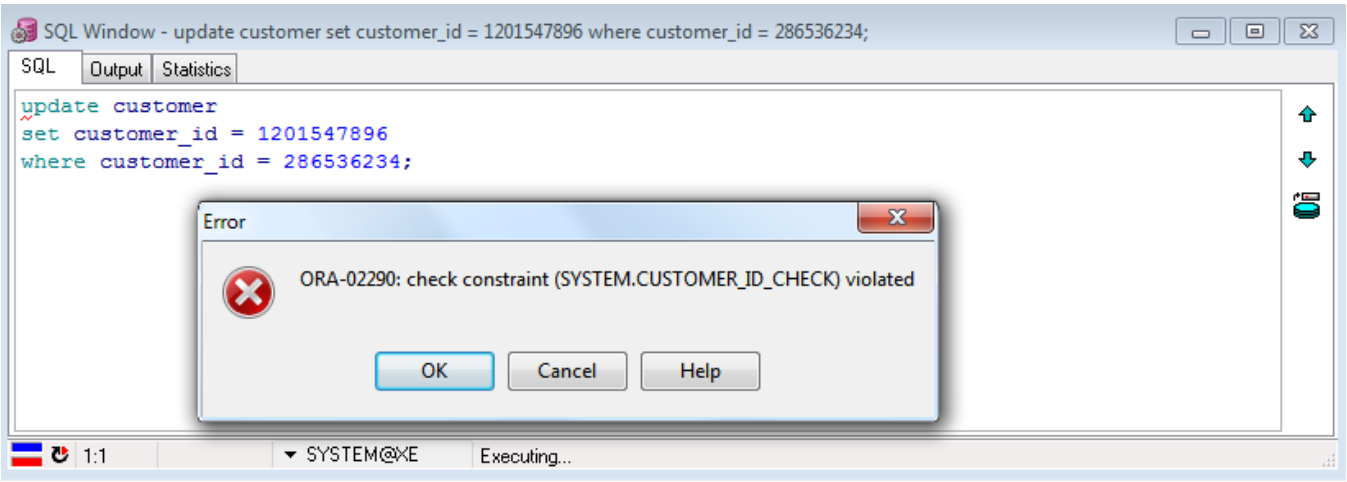
As we can see, the record has is still there despite the deletion, thanks to the rollback, just as expected.

5. Constraints

We define a new constraint, which checks that the `customer_ID` has 9 digits from 0 to 9, or in other words, that the `customer_ID` is strictly below 1000000000 (the lowest number with over 9 digits):



And now we update a customer's ID with an illegal Id (1201547896, which has 10 digits):



And as expected, the update is refused, because the constraint `customer_ID_check` was violated.

6. Views

1. Two views

The first view, `current_customers_policies`, shows the ongoing policies of all customers:

```
create view current_customers_policies as
select * from customers_policy
where end_date > sysdate;
```

The second view, `accounts`, shows the customer record information at a glance: ID, full name, number of subscribed policies and total price to pay for the month:

```
create view accounts as
select customer.customer_id, customer.first_name, customer.last_name,
       count(*) as number_of_policies, sum(price) as total_monthly_price
from policy
     join customers_policy on customers_policy.policy_id = policy.policy_id
     join customer on customer.customer_id = customers_policy.customer_id
group by customer.customer_id, customer.first_name, customer.last_name;
```

2. Two queries:

For the first view, `current_customers_policies`, this query outputs the number of ongoing policies of a particular customer whose ID is 250875791:

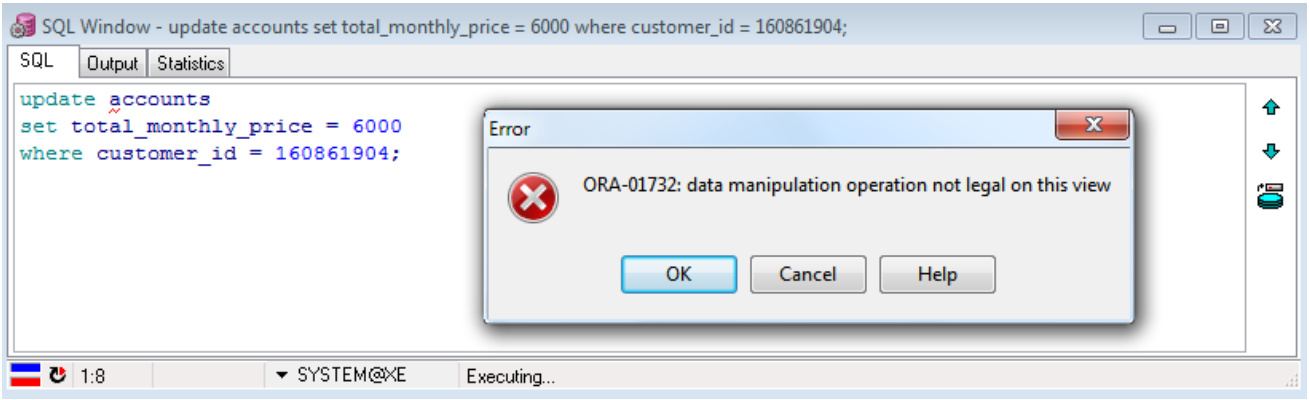
```
select * from current_customers_policies
where customer_id = 250875791;
```

And for the second view, `accounts`, this query displays the number of customers, of customers' policies, the total income and the average price for each customers' policy:

```
select sum(customer_id) as customers, sum(number_of_policies) as policies,
       sum(total_monthly_price) as income,
       sum(total_monthly_price)/sum(number_of_policies) as average_policy_cost
       from accounts;
```

3. View error

We attempt to update the total price for a given customer in the `accounts` view.



Of course, this would break the data integrity since a view's data is not stored but is just a saved query, and we cannot update the total directly but we have to update the price of the policy type itself.

Level 3 – PL (Programming Language) and Features

1. Two procedures / functions

1. p_updatePropertyPrice: This procedure updates the price of any given property insurance policy type. It takes as parameters the base price, which is the minimum price for all policies, and the ID of the policy for which the price should be updated. It then calculates the final price for the property policy from the attributes in the property table (according to the type of property, its age, and its value) and updates it accordingly in the policy table.

```
create or replace procedure p_updatePropertyPrice(i_basePrice_nr NUMBER,
i_policyID_nr NUMBER) is
  r_prop property%ROWTYPE;
  v_propFactor_nr NUMBER;
  v_ageOfProp_nr NUMBER;
  v_propValue_nr NUMBER;
  v_resultPrice_nr NUMBER;
begin
  --Load record
  select * into r_prop from property where policy_ID = i_policyID_nr;
  --Set type factor according to property code
  case r_prop.property_code
    when 1 then
      v_propFactor_nr := 1.5;
    when 2 then
      v_propFactor_nr := 2;
    when 3 then
      v_propFactor_nr := 2.5;
    when 4 then
      v_propFactor_nr := 3.5;
    when 5 then
      v_propFactor_nr := 5;
    else
      v_propFactor_nr := 1;
  end case;
  --Set age of property
  v_ageOfProp_nr := r_prop.age_of_property;
  --Set property value
  v_propValue_nr := r_prop.property_value;
  --Calculate resPrice
  v_resultPrice_nr := i_basePrice_nr + v_propFactor_nr*v_ageOfProp_nr*5 +
v_propValue_nr/10000;*
  --Update record
  update policy
  set price = v_resultPrice_nr
  where policy_id = i_policyID_nr;
end p_updatePropertyPrice;
```

2. f_sumForCustomer_Nr: This function returns the total price of all the policies of a particular customer ongoing at a given date. The parameters are the customer's ID and the said date. A cursor then loops over the customer's policies and sums the actual prices (policy type price multiplied by the customer's bonus), and then returns the sum.

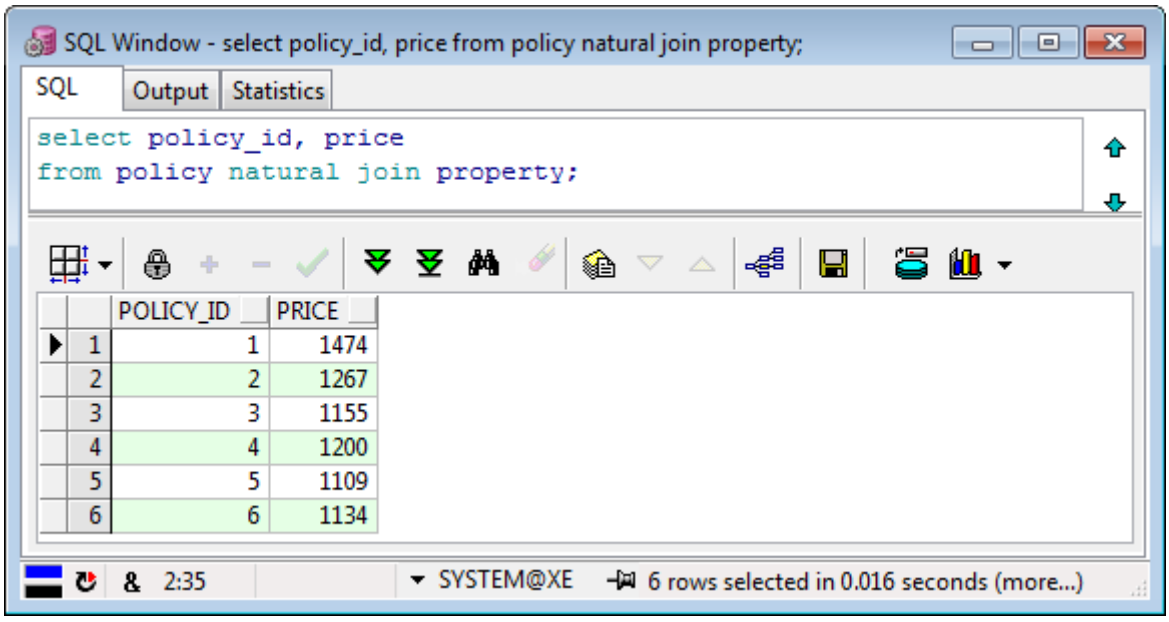
```
create or replace function f_sumForCustomer_Nr(i_customerID_nr NUMBER,
v_date_dt DATE)
return NUMBER
is
  cursor c_custPolicy is
    select * from policy natural join customers_policy
    where customer_ID = i_customerID_nr;
  r_custPolicy c_custPolicy%ROWTYPE;
  v_sumPrices_nr NUMBER := 0;
begin
  --Load cursor
  open c_custPolicy;
  loop
    --Load record
    fetch c_custPolicy into r_custPolicy;
    exit when c_custPolicy%NOTFOUND;
    --If policy date checks out
    if r_custPolicy.start_date < v_date_dt and r_custPolicy.end_date >
v_date_dt then
      --Compute price of actual policy using price and bonus and add to sum
      v_sumPrices_nr := v_sumPrices_nr + r_custPolicy.price * (1 - 0.05 *
r_custPolicy.bonus);
    end if;
  end loop;
  close c custPolicy;
  return(v_sumPrices_nr);
end f_sumForCustomer_Nr;
```

2. Two Programs for the above procedures/functions

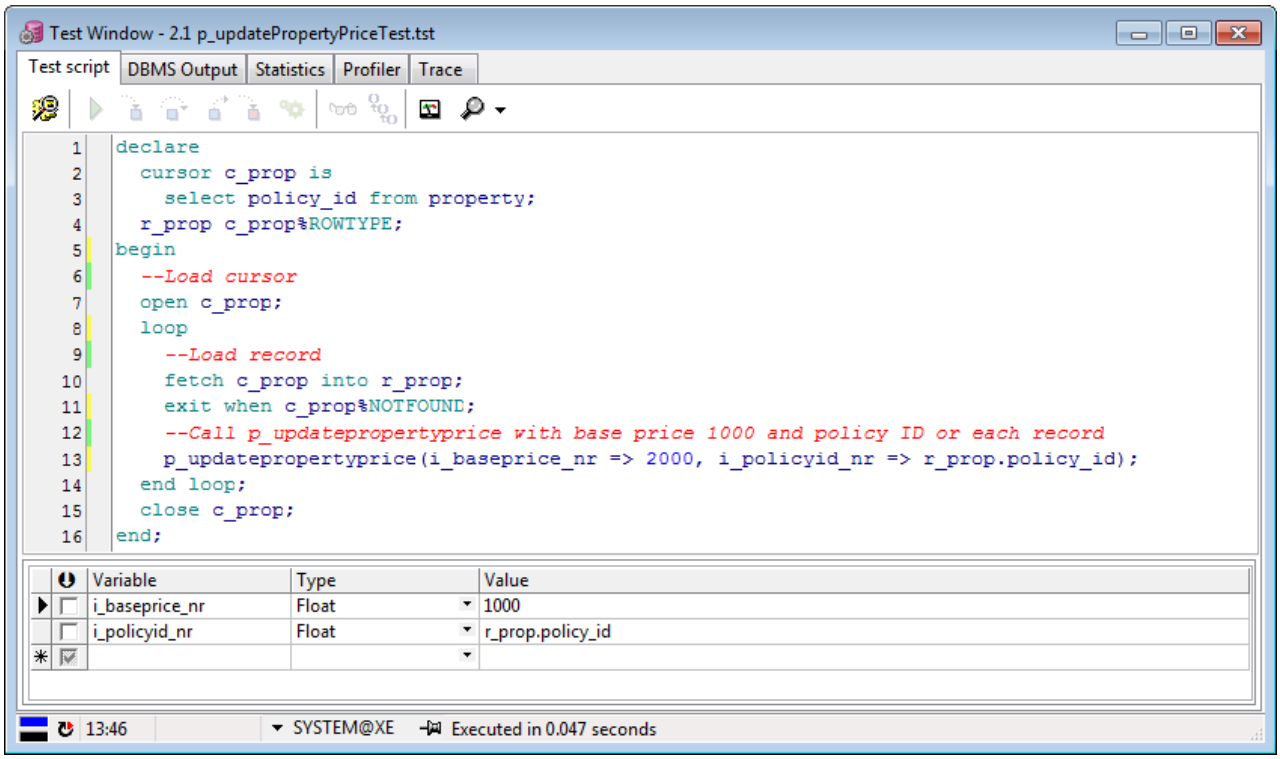
1. This program updates the prices of all the property policies using the first procedure above (1.1). In order to do this, it uses a cursor that goes through all the rows from the `property` table and calls the procedure to update the relevant row in the `policy` table.

```
declare
  cursor c_prop is
    select policy_id from property;
  r_prop c_prop%ROWTYPE;
begin
  --Load cursor
  open c_prop;
  loop
    --Load record
    fetch c_prop into r_prop;
    exit when c_prop%NOTFOUND;
    --Call p_updatepropertyprice with base price 1000 and policy ID or each
record
    p_updatepropertyprice(i_baseprice_nr => 1000,
    i_policyid_nr => r_prop.policy_id);
  end loop;
  close c_prop;
end;
```

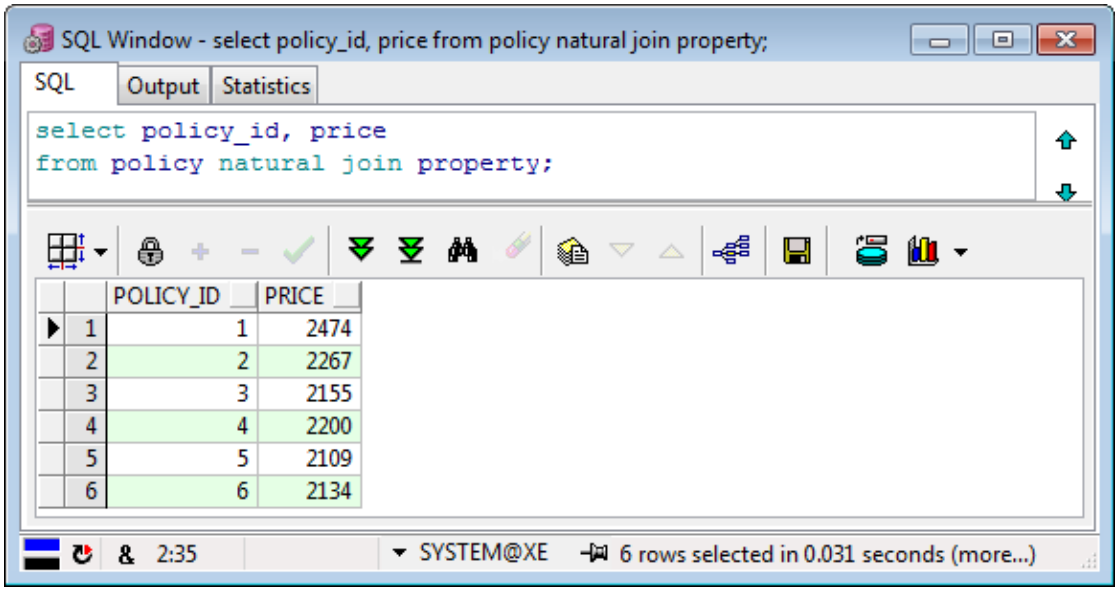
Let us check: first we query the database before the changes:



Then we run the procedure and commit:



And finally we query again with the same query and notice the updated results:

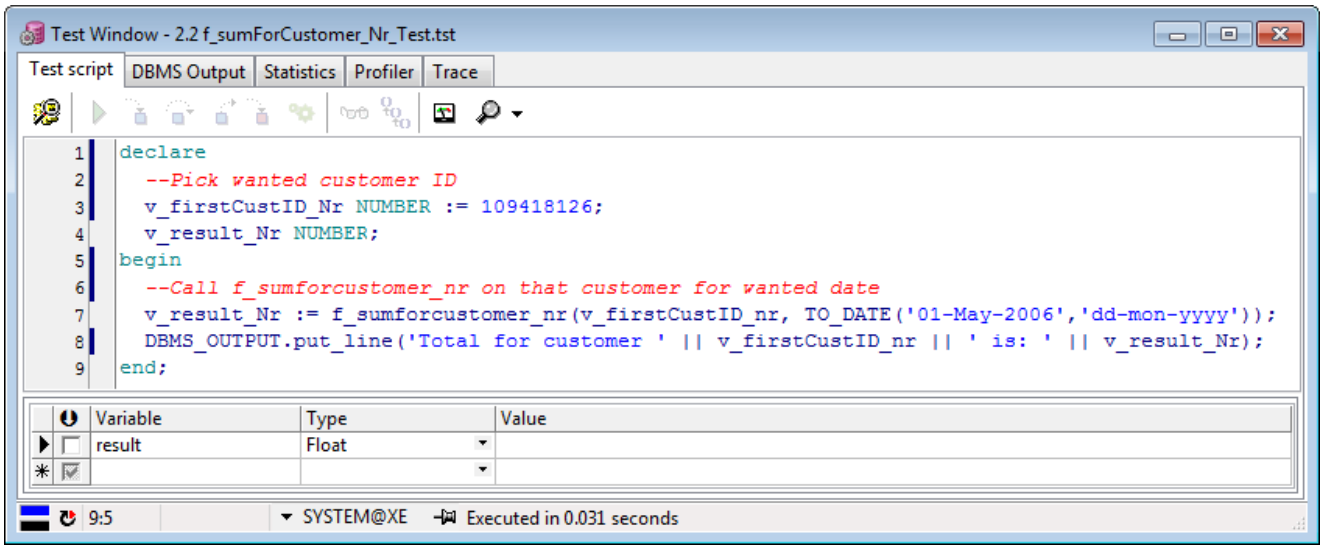


All as expected.

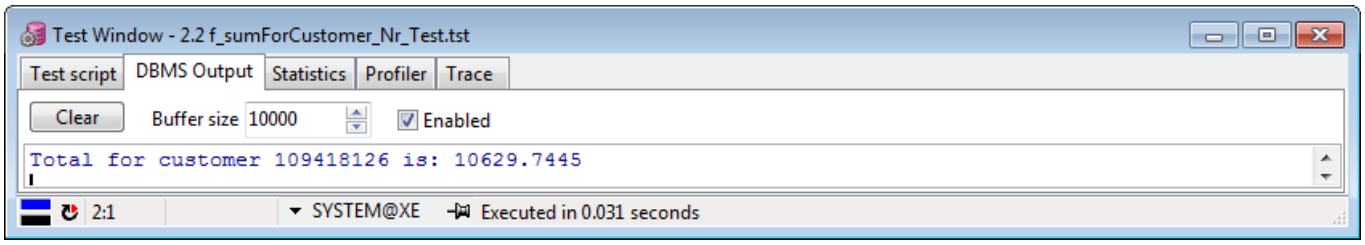
2. This program outputs the total price to pay for the `customer_id` 109418126 as of 01/05/2006.

```
declare
  --Pick wanted customer ID
  v_firstCustID_Nr NUMBER := 109418126;
  v_result_Nr NUMBER;
begin
  --Call f_sumforcustomer_nr on that customer for wanted date
  v_result_Nr := f_sumforcustomer_nr(v_firstCustID_nr, TO_DATE('01-May-
2006','dd-mon-yyyy'));
  DBMS_OUTPUT.put_line('Total for customer ' || v_firstCustID_nr || ' is: ' ||
v_result_Nr);
end;
```

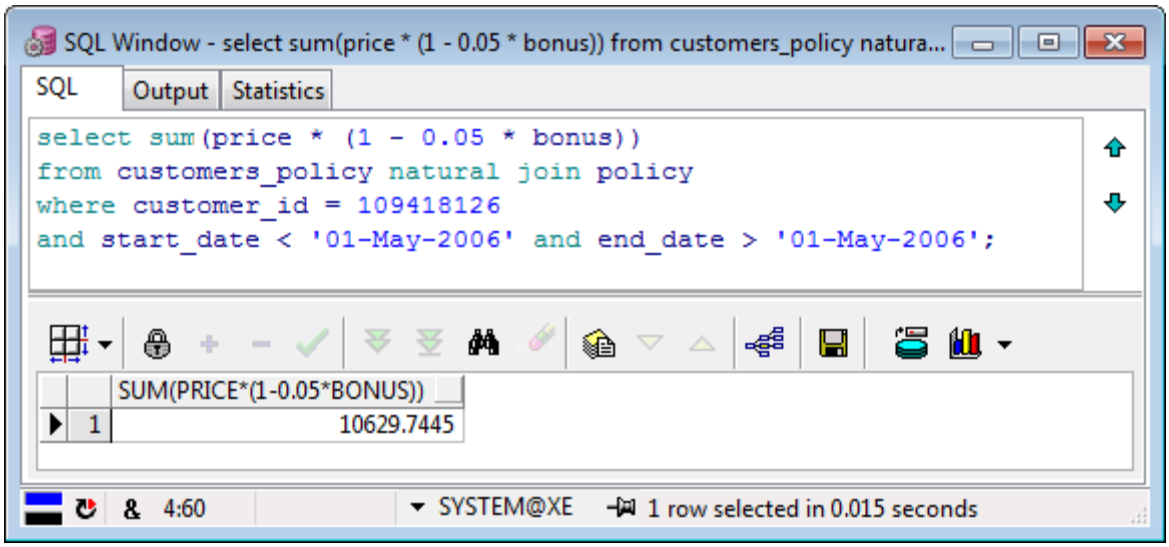
Let us check: we run the function:



Then we read the DBMS output tab:



And finally we can query the database to verify that the result checks out:



All good.

3. Two triggers

1. pol_checkDate acts like a constraint which, instead of each insert into `customers_policy`, checks whether the `start_date` is more than a month later than today, in which case it calls a `raise_application_error` with an informative error message (unlike usual constraints) and rejects the insert query:

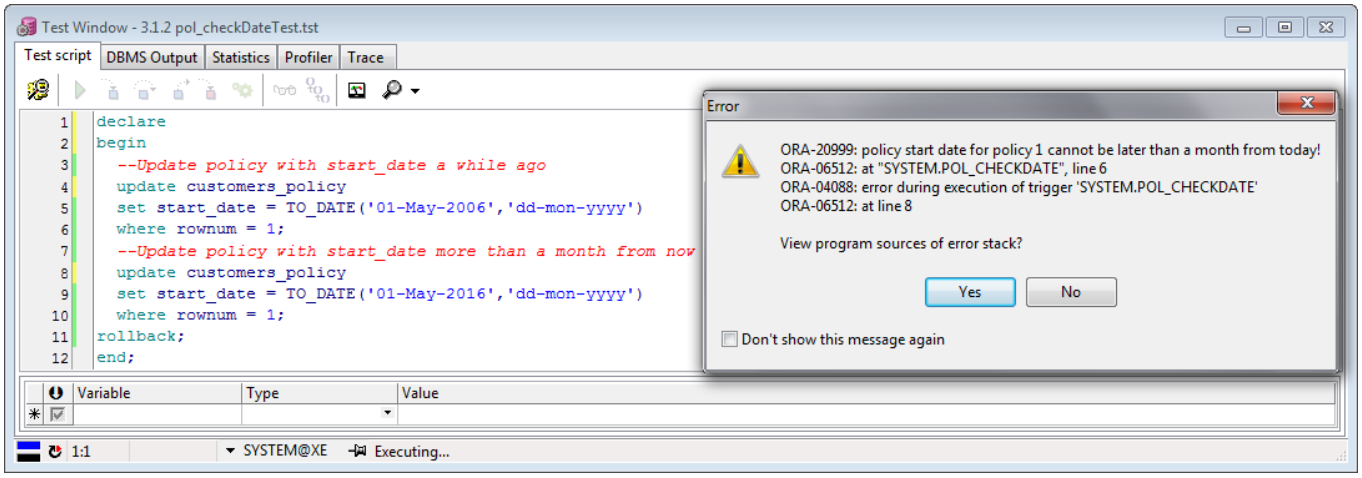
:

```
create or replace trigger pol_checkDate
  before insert or update on customers_policy
  for each row
declare
  v_error_tx VARCHAR2(2000);
begin
  --If start date of policy more than a month from now
  if :new.start_date > sysdate + 30 then
    v_error_tx := 'policy start date for policy ' || :new.customers_policy_id
|| ' cannot be later than a month from today!';
    --Reject query and print error message
    raise_application_error(-20999, v_error_tx);
  end if;
end property_setPrice;
```

And here is a program that checks the constraints by entering one legal statement and an illegal one (as of this year, 2013):

```
declare
begin
  --Update policy with start_date a while ago
  update customers_policy
  set start_date = TO_DATE('01-May-2006','dd-mon-yyyy')
  where rownum = 1;
  --Update policy with start_date more than a month from now
  update customers_policy
  set start_date = TO_DATE('01-May-2016','dd-mon-yyyy')
  where rownum = 1;
end;
```

Let's run the above program:



The program complained about the update at line 8, as expected.

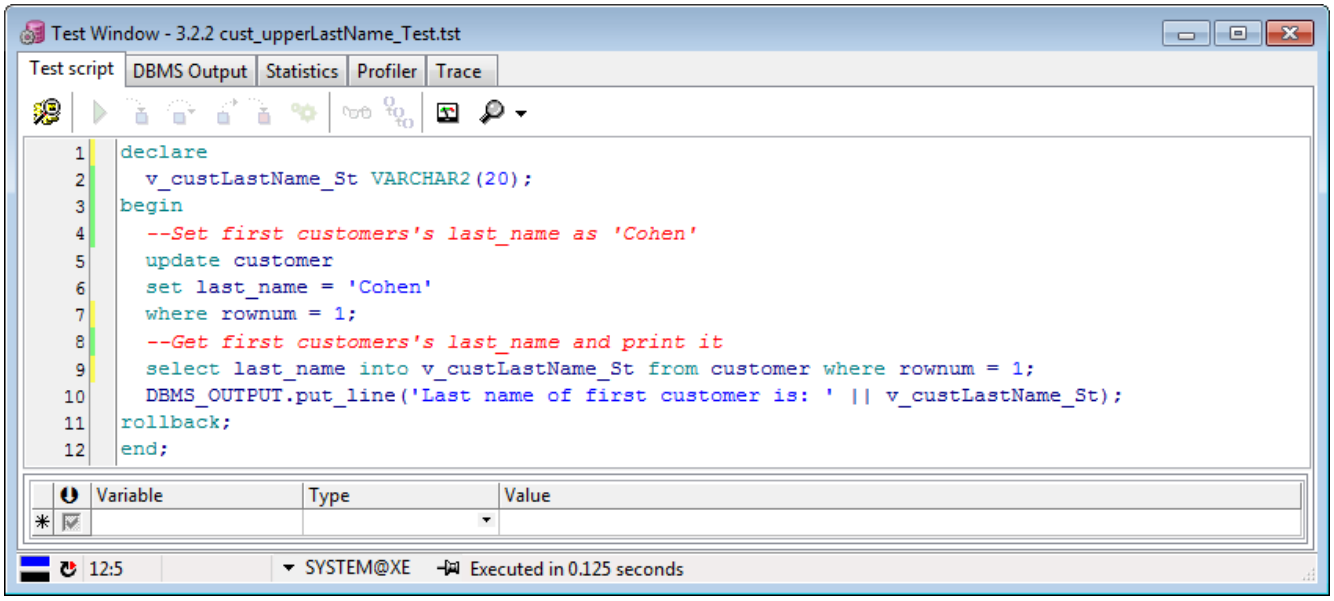
2. cust_upperLastName forces the `last_name` of each new insert or update into `customer` to be uppercase, as is often the case in administration offices:

```
create or replace trigger cust_upperLastName
  before insert or update on customer
  for each row
begin
  --Replace last_name with same string but uppercase characters
  :new.last_name := upper(:new.last_name);
end cust_upperLastName;
```

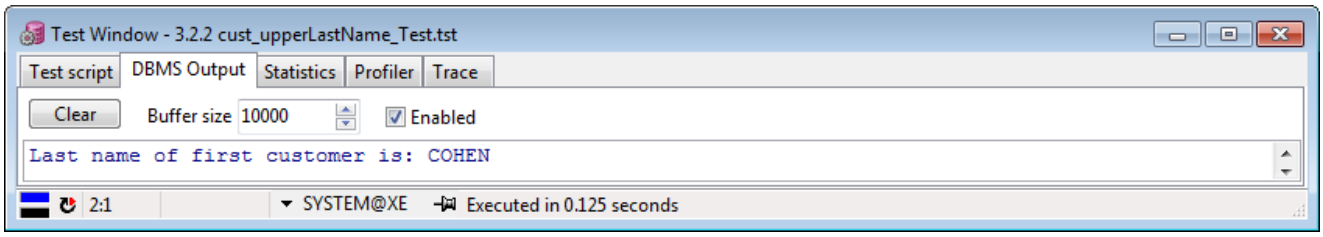
This program tests the above trigger by updating the `last_name` of the first customer with the name 'Cohen' and then outputs the `last_name` as it appears in the database after the update.

```
declare
  v_custLastName_St VARCHAR2(20);
begin
  --Set first customers's last_name as 'Cohen'
  update customer
  set last_name = 'Cohen'
  where rownum = 1;
  --Get first customers's last_name and print it
  select last_name into v_custLastName_St from customer where rownum = 1;
  DBMS_OUTPUT.put_line('Last name of first customer is: ' ||
v_custLastName_St);
rollback;
end;
```

Now let us run the above program:



Let's check the updated `last_name` using the DBMS output tab:



Checks out.

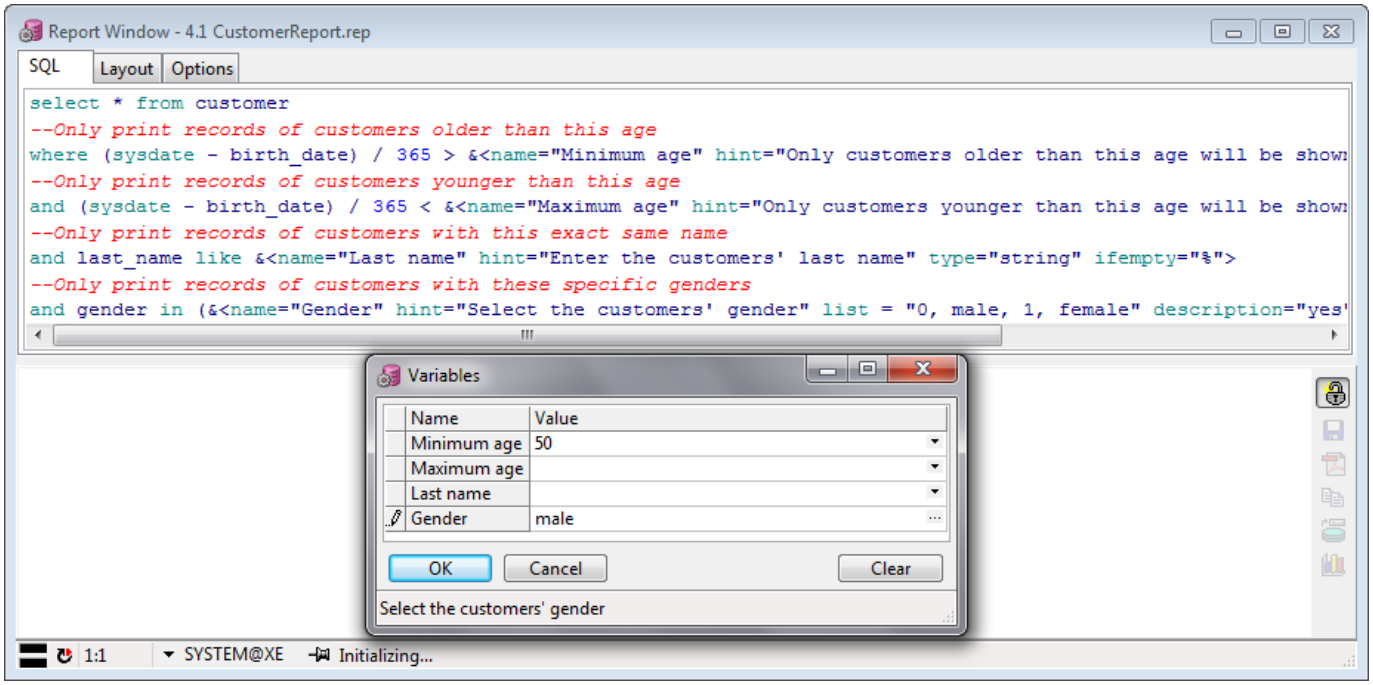
4. Report and Substitution Variables

1. Customers report

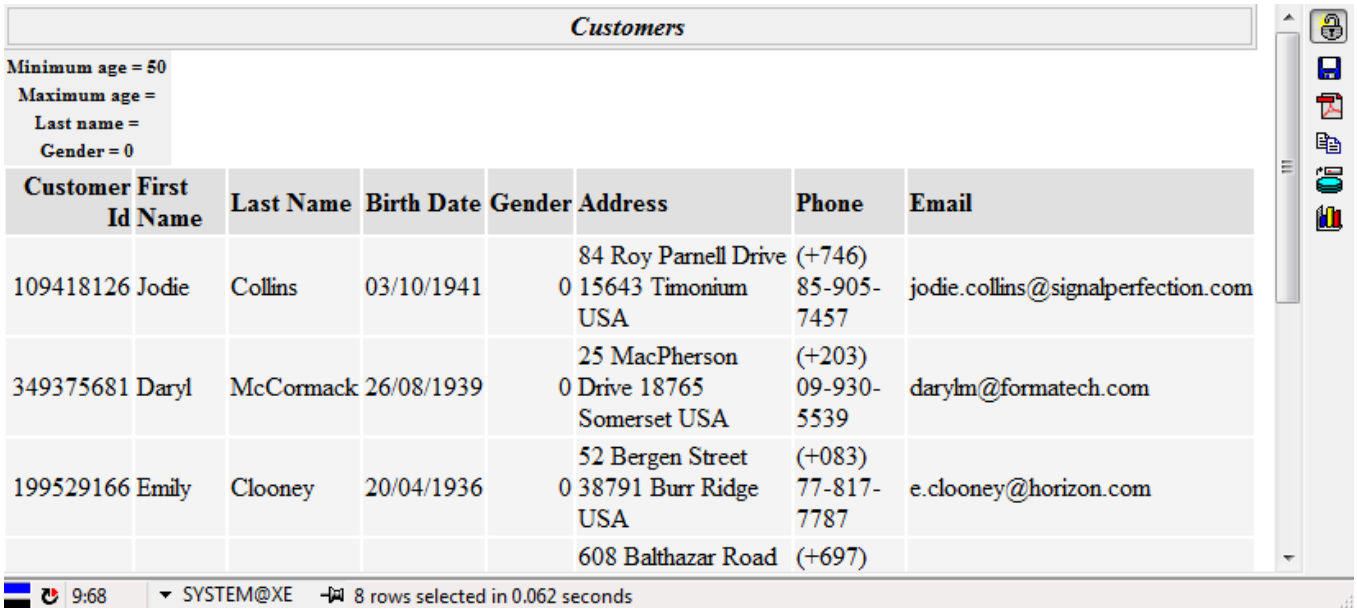
This report presents the customer information filtered by minimum or maximum age (we can also define a range this way), by last name and by gender (using an optional list):

```
select * from customer
--Only print records of customers older than this age
where (sysdate - birth_date) / 365 > &<name="Minimum age"
  hint="Only customers older than this age will be shown" type="integer"
  ifempty="0">
--Only print records of customers younger than this age
and (sysdate - birth_date) / 365 < &<name="Maximum age"
  hint="Only customers younger than this age will be shown" type="integer"
  ifempty="200">
--Only print records of customers with this exact same name
and last_name like &<name="Last name" hint="Enter the customers' last name"
  type="string" ifempty="%">
--Only print records of customers with these specific genders
and gender in (&<name="Gender" hint="Select the customers' gender"
  list   =   "0,   male,   1,   female"   description="yes"   multiselect="yes"
ifempty="0 ,1">)
```

Let's run it:



And we get this:



Customers

Minimum age = 50
Maximum age =
Last name =
Gender = 0

Customer Id	First Name	Last Name	Birth Date	Gender	Address	Phone	Email
109418126	Jodie	Collins	03/10/1941	0	84 Roy Parnell Drive 15643 Timonium USA	(+746) 85-905-7457	jodie.collins@signalperfection.com
349375681	Daryl	McCormack	26/08/1939	0	25 MacPherson Drive 18765 Somerset USA	(+203) 09-930-5539	darylm@formatech.com
199529166	Emily	Clooney	20/04/1936	0	52 Bergen Street 38791 Burr Ridge USA	(+083) 77-817-7787	e.clooney@horizon.com
					608 Balthazar Road	(+697)	

9:68 SYSTEM@XE 8 rows selected in 0.062 seconds

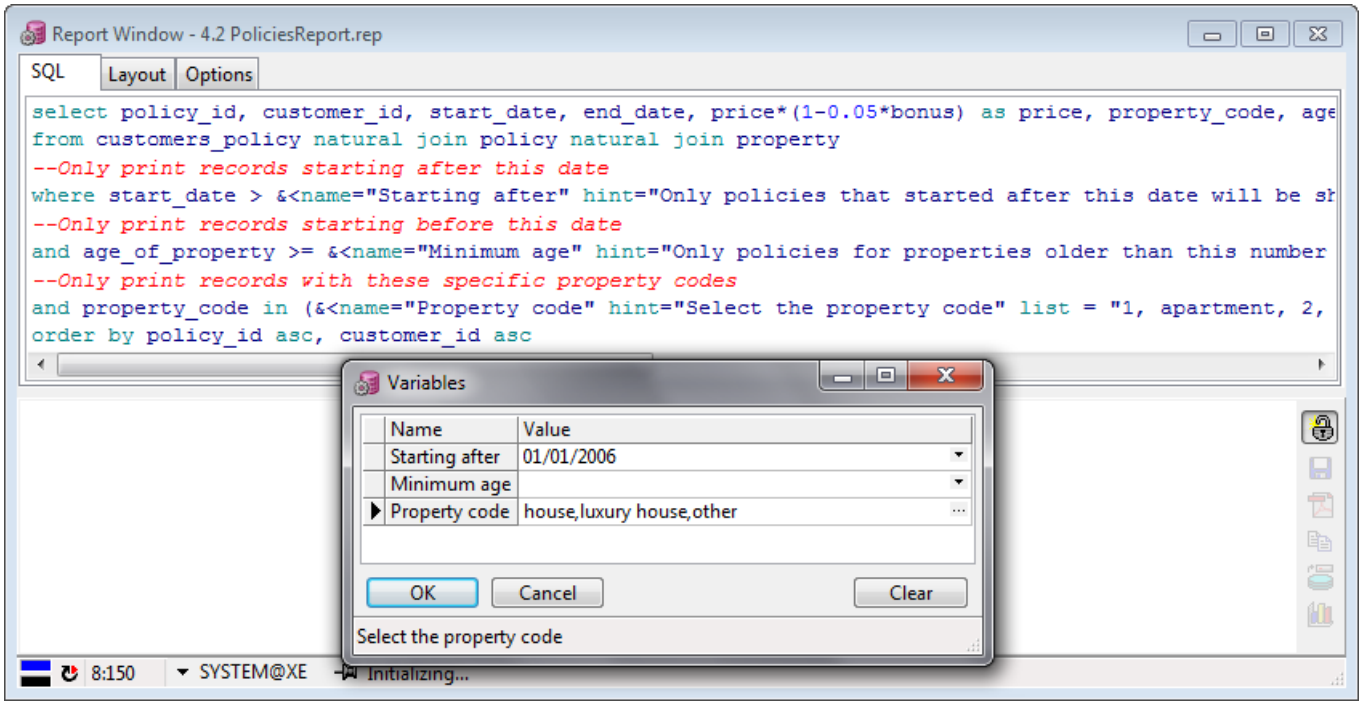
Excellent.

2. Policies report

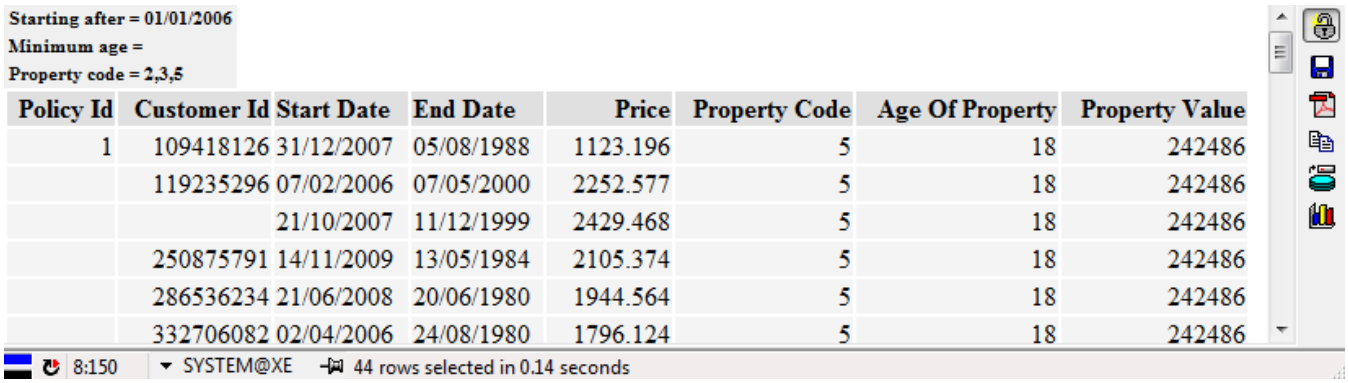
This report presents various information about actual implemented property insurance policies, filtered by minimum start date, age of property and property code (using an optional list). It also offers a layout with ordering and block.

```
select policy_id, customer_id, start_date, end_date, price*(1-0.05*bonus) as
price,
  property_code, age_of_property, property_value
from customers_policy natural join policy natural join property
--Only print records starting after this date
where start_date > &<name="Starting after"
  hint="Only policies that started after this date will be shown"
  type="date" ifempty="01/01/1900">
--Only print records starting before this date
and age_of_property >= &<name="Minimum age"
  hint="Only policies for properties older than this number (in years) will be
shown"
  type="integer" ifempty="0">
--Only print records with these specific property codes
and property_code in (&<name="Property code" hint="Select the property code"
  list = "1, apartment, 2, house, 3, luxury house, 4, building, 5, other"
  description="yes" multiselect="yes" ifempty="1, 2, 3, 4, 5">)
order by policy_id asc, customer_id asc
```

Let us run it:



And this is the result:



Policy Id	Customer Id	Start Date	End Date	Price	Property Code	Age Of Property	Property Value
1	109418126	31/12/2007	05/08/1988	1123.196	5	18	242486
	119235296	07/02/2006	07/05/2000	2252.577	5	18	242486
		21/10/2007	11/12/1999	2429.468	5	18	242486
	250875791	14/11/2009	13/05/1984	2105.374	5	18	242486
	286536234	21/06/2008	20/06/1980	1944.564	5	18	242486
	332706082	02/04/2006	24/08/1980	1796.124	5	18	242486

You can notice that we used the layout tab for breaks and tweaked categorization. Perfect.

Level 4 – Graphical User Interface (GUI)

1. Description

We wrote a graphical user interface (GUI) in ASP.NET with C#.NET backend code.

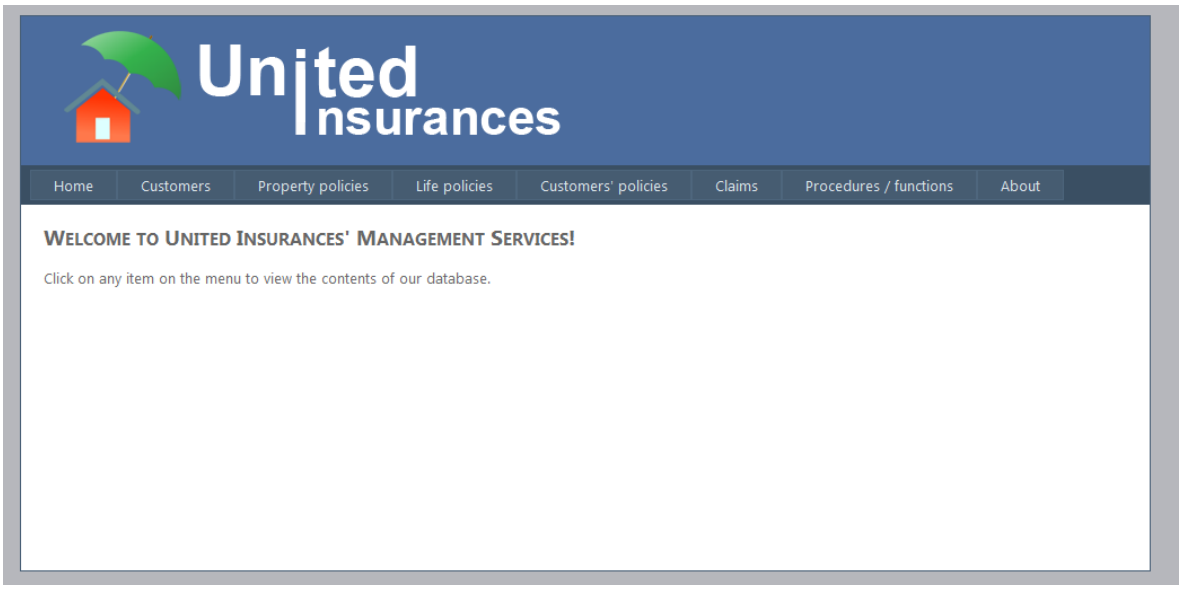
For the interface, we used:

- The default layout with slightly modified CSS and a custom logo made in Photoshop.
- Gridviews for showing the table content and selecting rows, and DetailsViews for editing the row content and handling operations, all binded on SqlDataSources with select, insert, update and delete abilities. These may also be used to demonstrate constraints and triggers.
- Tables, Labels, Textboxes, DropdownLists, a Calendar and Buttons for the form showing the behavior of the function and procedure.

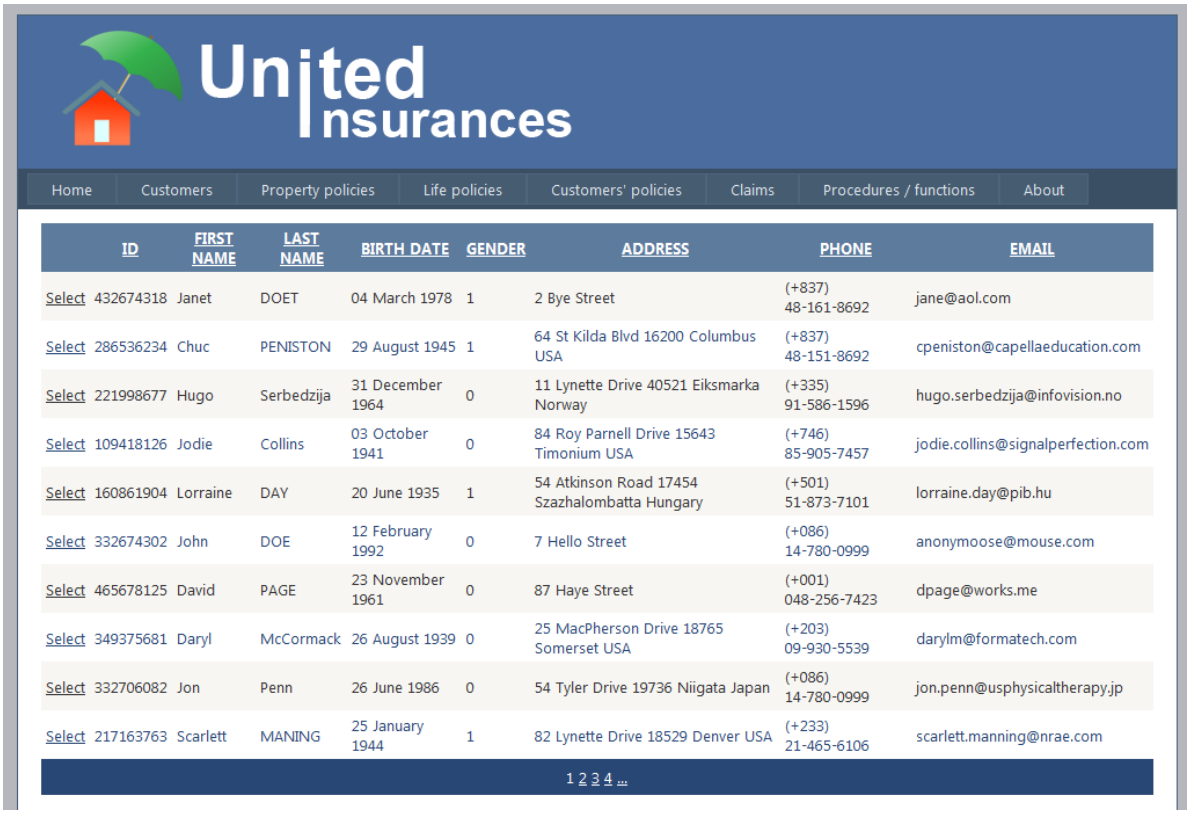
For the backend, we wrote very simple C# with straightforward functions for extremely quick development. This resulted in possible instability, vulnerability (there are two queries for which I have not cleaned the input, exposing us to XSS) and lack of extensibility and reusability.
The development was not very easy either (it took three days and quite a bit of headache) but it is relatively reliable, as far as we have tested.

2. Pictures of each screen

1. ~/Home.aspx

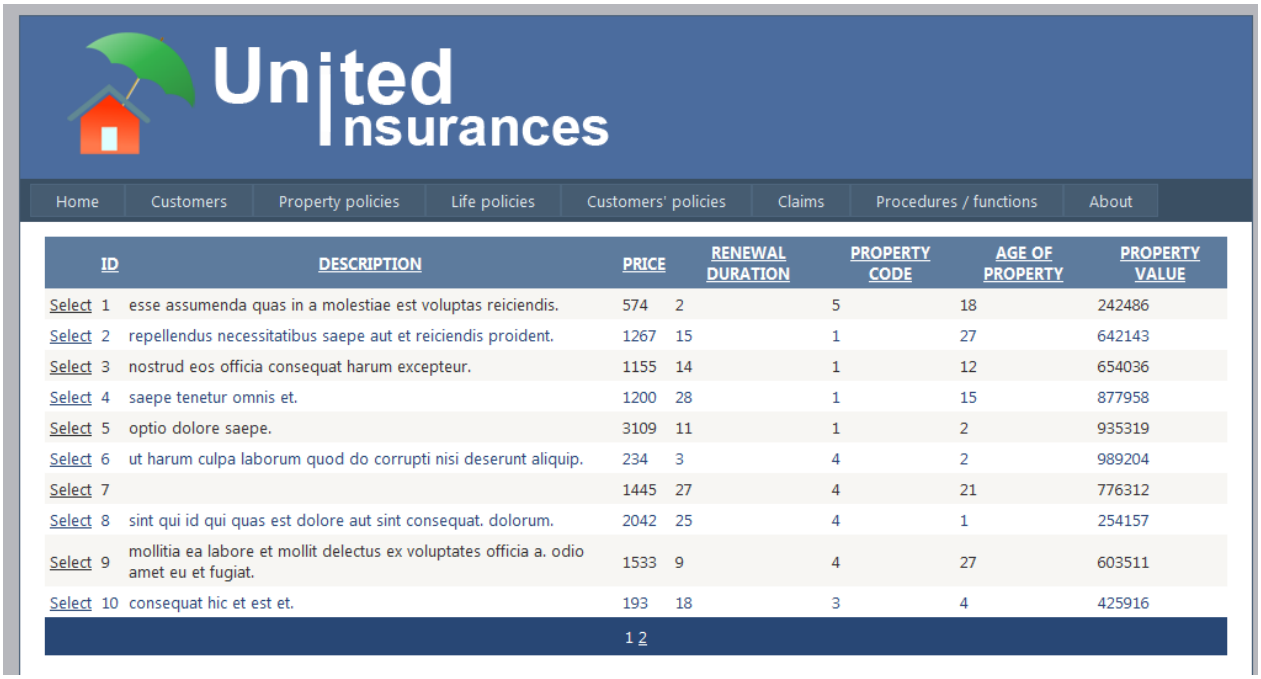


2. ~/Customers.aspx



	ID	FIRST NAME	LAST NAME	BIRTH DATE	GENDER	ADDRESS	PHONE	EMAIL
Select	432674318	Janet	DOET	04 March 1978	1	2 Bye Street	(+837) 48-161-8692	jane@aol.com
Select	286536234	Chuc	PENISTON	29 August 1945	1	64 St Kilda Blvd 16200 Columbus USA	(+837) 48-151-8692	cpeniston@capellaeducation.com
Select	221998677	Hugo	Serbedzija	31 December 1964	0	11 Lynette Drive 40521 Eiksmarka Norway	(+335) 91-586-1596	hugo.serbedzija@infovision.no
Select	109418126	Jodie	Collins	03 October 1941	0	84 Roy Parnell Drive 15643 Timonium USA	(+746) 85-905-7457	jodie.collins@signalperfection.com
Select	160861904	Lorraine	DAY	20 June 1935	1	54 Atkinson Road 17454 Szazhalombatta Hungary	(+501) 51-873-7101	lorraine.day@pib.hu
Select	332674302	John	DOE	12 February 1992	0	7 Hello Street	(+086) 14-780-0999	anonymoose@mouse.com
Select	465678125	David	PAGE	23 November 1961	0	87 Haye Street	(+001) 048-256-7423	dpage@works.me
Select	349375681	Daryl	McCormack	26 August 1939	0	25 MacPherson Drive 18765 Somerset USA	(+203) 09-930-5539	darylm@formatech.com
Select	332706082	Jon	Penn	26 June 1986	0	54 Tyler Drive 19736 Niigata Japan	(+086) 14-780-0999	jon.penn@usphysicaltherapy.jp
Select	217163763	Scarlett	MANING	25 January 1944	1	82 Lynette Drive 18529 Denver USA	(+233) 21-465-6106	scarlett.manning@nrae.com

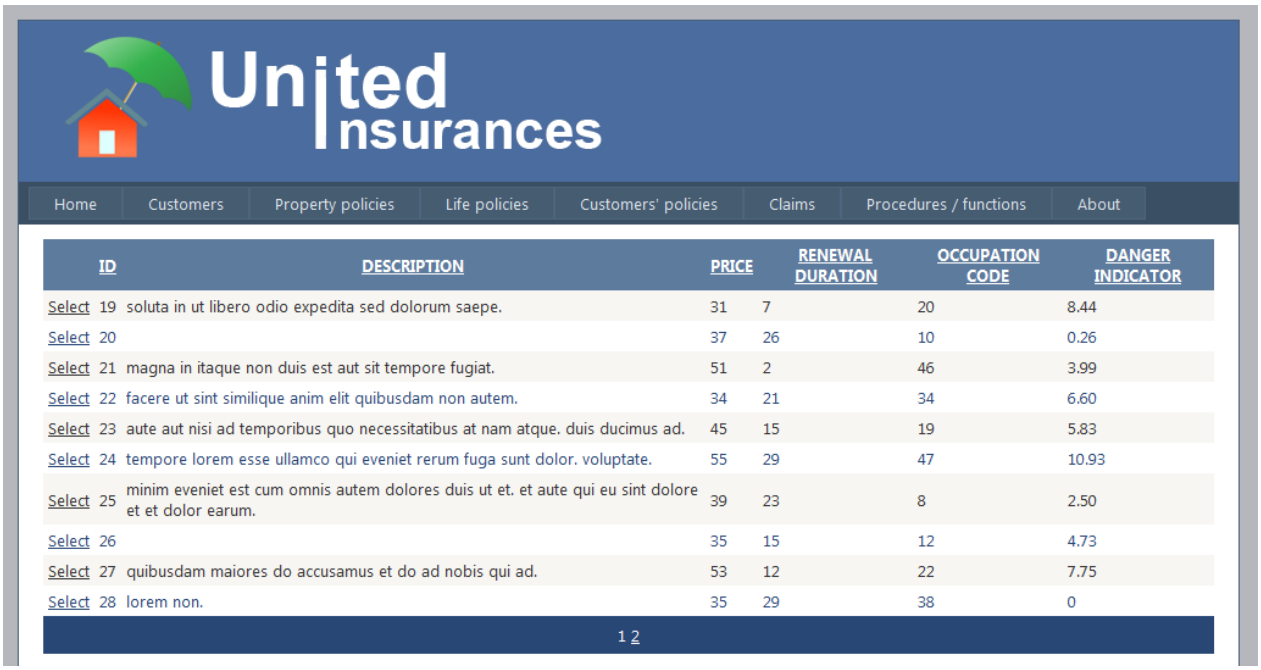
3. ~/PropertyPolicies.aspx



	ID	DESCRIPTION	PRICE	RENEWAL DURATION	PROPERTY CODE	AGE OF PROPERTY	PROPERTY VALUE
Select	1	esse assumenda quas in a molestiae est voluptas reiciendis.	574	2	5	18	242486
Select	2	repellendus necessitatibus saepe aut et reiciendis proident.	1267	15	1	27	642143
Select	3	nostrud eos officia consequat harum excepteur.	1155	14	1	12	654036
Select	4	saepe tenetur omnis et.	1200	28	1	15	877958
Select	5	optio dolore saepe.	3109	11	1	2	935319
Select	6	ut harum culpa laborum quod do corrupti nisi deserunt aliquip.	234	3	4	2	989204
Select	7		1445	27	4	21	776312
Select	8	sint qui id qui quas est dolore aut sint consequat. dolorum.	2042	25	4	1	254157
Select	9	mollitia ea labore et mollit delectus ex voluptates officia a. odio amet eu et fugiat.	1533	9	4	27	603511
Select	10	consequat hic et est et.	193	18	3	4	425916

1 2

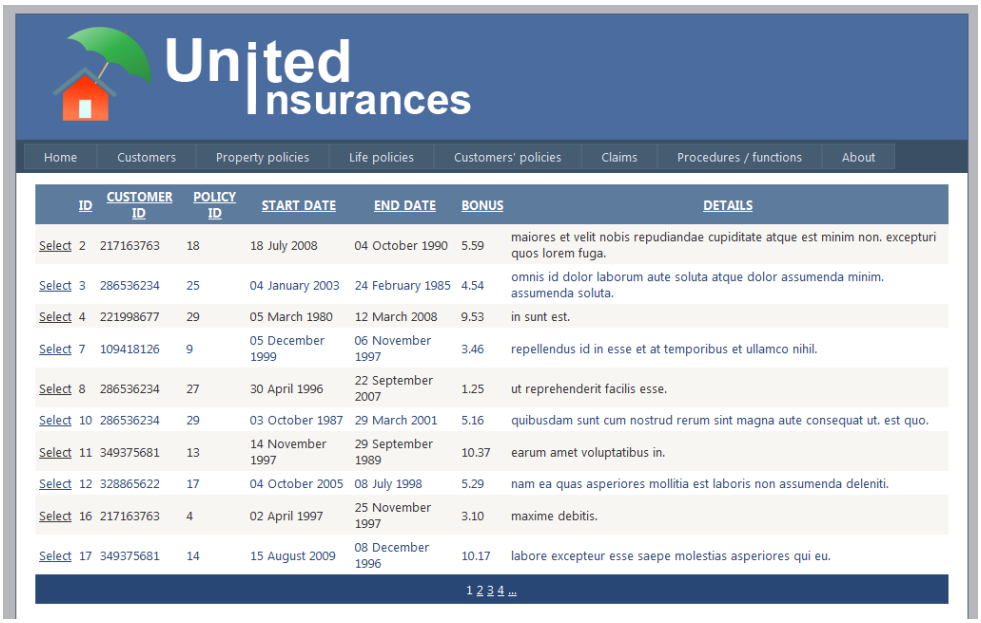
4. ~/LifePolicies.aspx



	ID	DESCRIPTION	PRICE	RENEWAL DURATION	OCCUPATION CODE	DANGER INDICATOR
Select	19	soluta in ut libero odio expedita sed dolorum saepe.	31	7	20	8.44
Select	20		37	26	10	0.26
Select	21	magna in itaque non duis est aut sit tempore fugiat.	51	2	46	3.99
Select	22	facere ut sint similique anim elit quibusdam non autem.	34	21	34	6.60
Select	23	aute aut nisi ad temporibus quo necessitatibus at nam atque. duis ducimus ad.	45	15	19	5.83
Select	24	tempore lorem esse ullamco qui eveniet rerum fuga sunt dolor. voluptate.	55	29	47	10.93
Select	25	minim eveniet est cum omnis autem dolores duis ut et. et aute qui eu sint dolore et et dolor earum.	39	23	8	2.50
Select	26		35	15	12	4.73
Select	27	quibusdam maiores do accusamus et do ad nobis qui ad.	53	12	22	7.75
Select	28	lorem non.	35	29	38	0

1 2

5. ~/CustomersPolicies.aspx



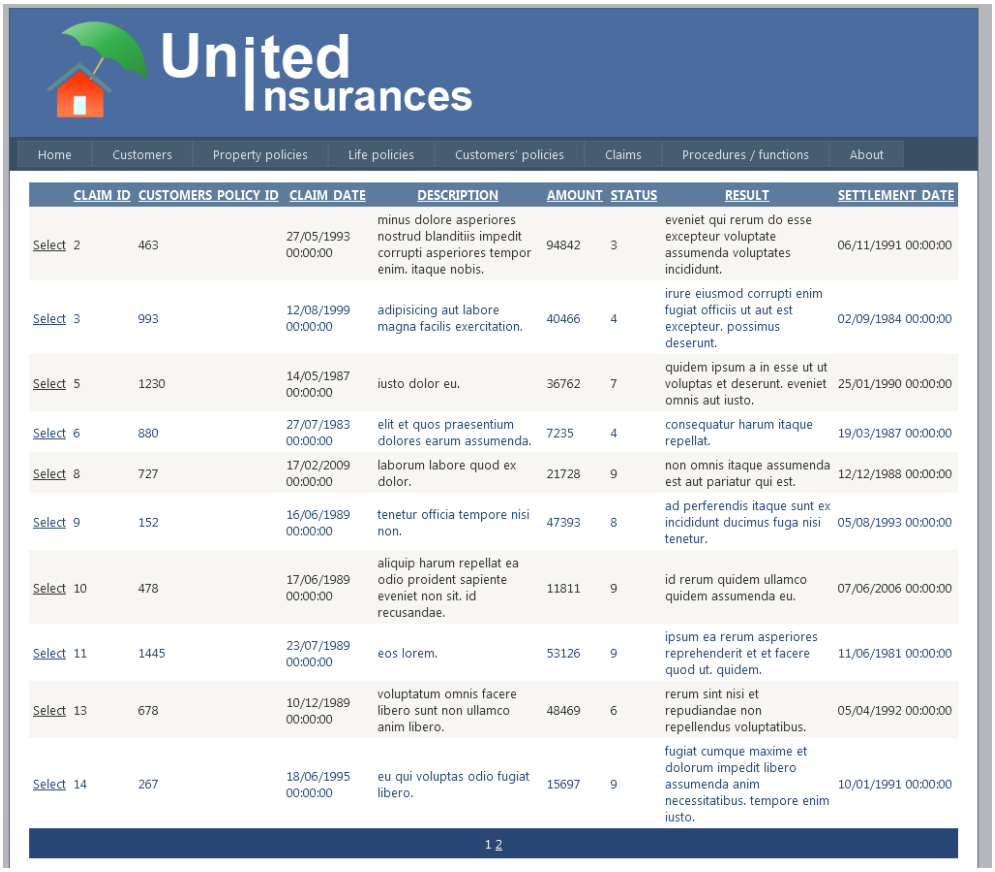
United Insurances

Home | Customers | Property policies | Life policies | Customers' policies | Claims | Procedures / functions | About

	ID	CUSTOMER ID	POLICY ID	START DATE	END DATE	BONUS	DETAILS
Select	2	217163763	18	18 July 2008	04 October 1990	5.59	maiores et velit nobis repudiandae cupiditate atque est minim non. excepturi quos lorem fuga.
Select	3	286536234	25	04 January 2003	24 February 1985	4.54	omnis id dolor laborum aute soluta atque dolor assumenda minim. assumenda soluta.
Select	4	221998677	29	05 March 1980	12 March 2008	9.53	in sunt est.
Select	7	109418126	9	05 December 1999	06 November 1997	3.46	repellendus id in esse et at temporibus et ullamco nihil.
Select	8	286536234	27	30 April 1996	22 September 2007	1.25	ut reprehenderit facilis esse.
Select	10	286536234	29	03 October 1987	29 March 2001	5.16	quibusdam sunt cum nostrud rerum sint magna aute consequat ut. est quo.
Select	11	349375681	13	14 November 1997	29 September 1989	10.37	earum amet voluptatibus in.
Select	12	328865622	17	04 October 2005	08 July 1998	5.29	nam ea quas asperiores mollitia est laboris non assumenda deleniti.
Select	16	217163763	4	02 April 1997	25 November 1997	3.10	maxime debitis.
Select	17	349375681	14	15 August 2009	08 December 1996	10.17	labore excepteur esse saepe molestias asperiores qui eu.

1 2 3 4 ...

6. ~/Claims.aspx



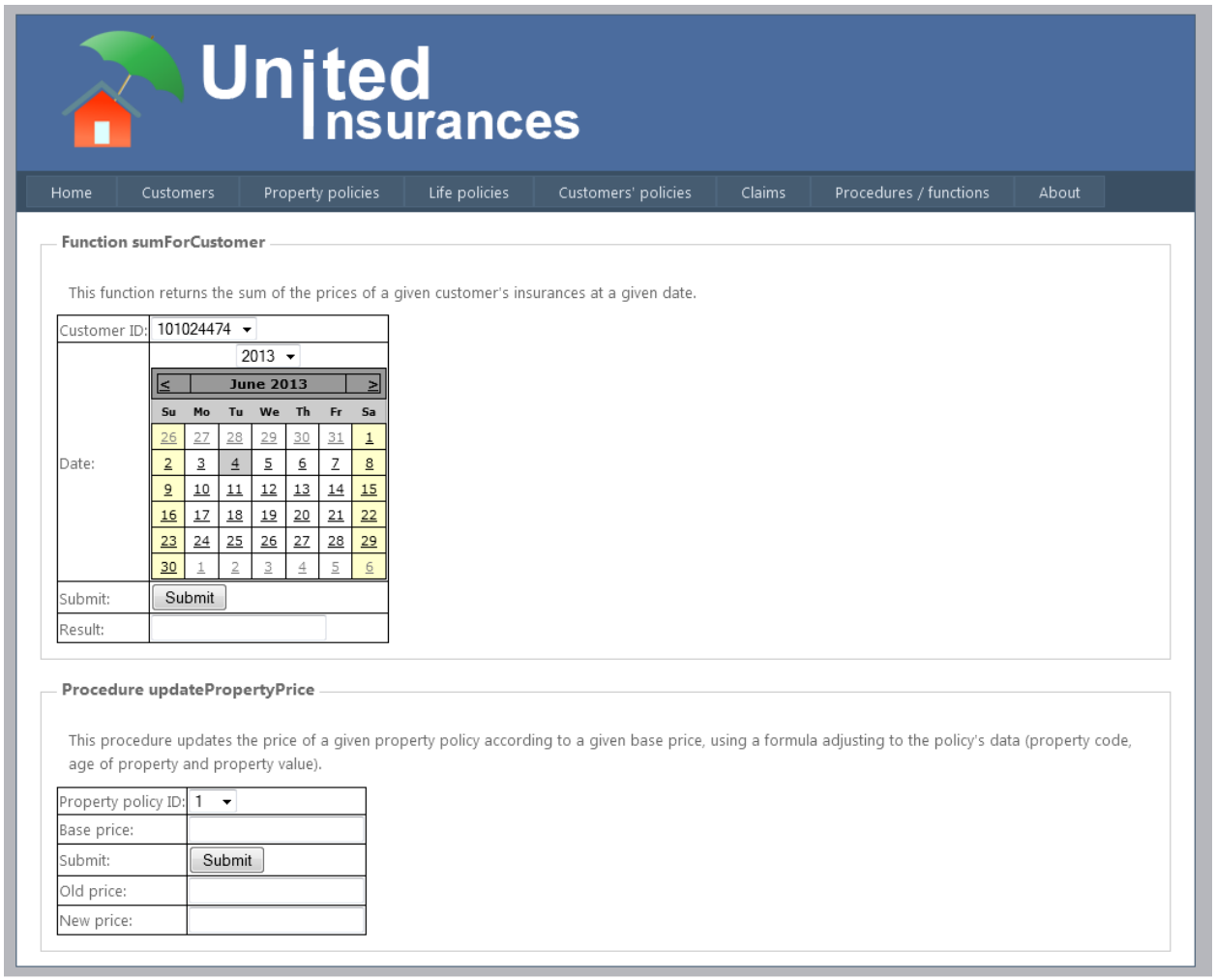
United Insurances

Home | Customers | Property policies | Life policies | Customers' policies | Claims | Procedures / functions | About

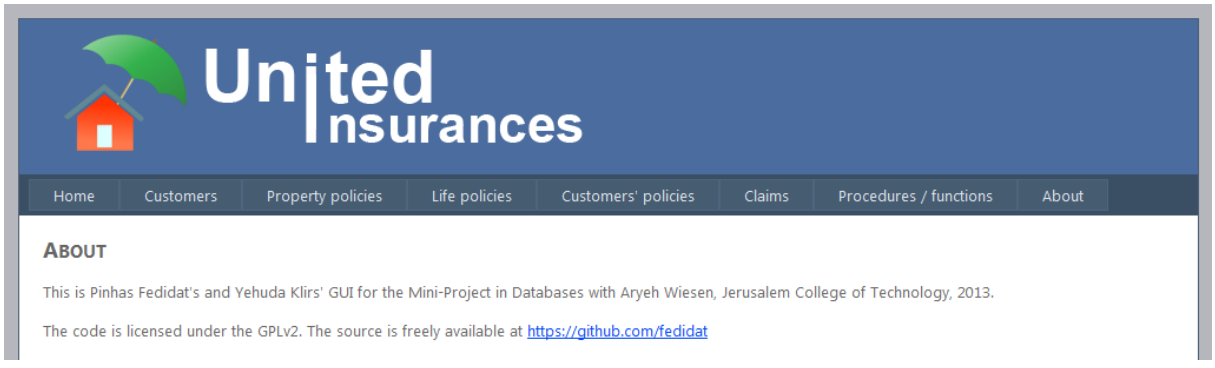
	CLAIM ID	CUSTOMERS POLICY ID	CLAIM DATE	DESCRIPTION	AMOUNT	STATUS	RESULT	SETTLEMENT DATE
Select	2	463	27/05/1993 00:00:00	minus dolore asperiores nostrud blanditiis impedit corrupti asperiores tempor enim. itaque nobis.	94842	3	eveniet qui rerum do esse excepteur voluptate assumenda voluptates incididunt.	06/11/1991 00:00:00
Select	3	993	12/08/1999 00:00:00	adipisicing aut labore magna facilis exercitation.	40466	4	irure eiusmod corrupti enim fugiat officiis ut aut est excepteur. possimus deserunt.	02/09/1984 00:00:00
Select	5	1230	14/05/1987 00:00:00	iusto dolor eu.	36762	7	quidem ipsum a in esse ut ut voluptas et deserunt. eveniet omnis aut iusto.	25/01/1990 00:00:00
Select	6	880	27/07/1983 00:00:00	elit et quos praesentium dolores earum assumenda.	7235	4	consequatur harum itaque repellat.	19/03/1987 00:00:00
Select	8	727	17/02/2009 00:00:00	laborum labore quod ex dolor.	21728	9	non omnis itaque assumenda est aut pariatur qui est.	12/12/1988 00:00:00
Select	9	152	16/06/1989 00:00:00	tenetur officia tempore nisi non.	47393	8	ad perferendis itaque sunt ex incididunt ducimus fuga nisi tenetur.	05/08/1993 00:00:00
Select	10	478	17/06/1989 00:00:00	aliquip harum repellat ea odio proident sapiente eveniet non sit. id recusandae.	11811	9	id rerum quidem ullamco quidem assumenda eu.	07/06/2006 00:00:00
Select	11	1445	23/07/1989 00:00:00	eos lorem.	53126	9	ipsum ea rerum asperiores reprehenderit et et facere quod ut. quidem.	11/06/1981 00:00:00
Select	13	678	10/12/1989 00:00:00	voluptatum omnis facere libero sunt non ullamco anim libero.	48469	6	rerum sint nisi et repudiandae non repellendus voluptatibus.	05/04/1992 00:00:00
Select	14	267	18/06/1995 00:00:00	eu qui voluptas odio fugiat libero.	15697	9	fugiat cumque maxime et dolorum impedit libero assumenda anim necessitatibus. tempore enim iusto.	10/01/1991 00:00:00

1 2

7. ~/Procfunc.aspx



8. "~/About.aspx



3. Demonstration

We will demonstrate the operations SELECT, INSERT, UPDATE, DELETE and the use of the procedure and of the function that we wrote in Level 3

1. SELECT

The page ~/PropertyPolicies.aspx shows the result of `select * from policy natural join property` in its GridView (from the sqlDataSource it is binded to, named OraInsurance), as shown below:

PropertyPolicies.aspx

Client Objects & Events | (No Events)

```
37  <asp:SqlDataSource ID="OraInsurance" runat="server"
38  ConnectionString="<%$ ConnectionStrings:OraInsuranceConnection %>"
39  ProviderName="<%$ ConnectionStrings:OraInsuranceConnection.ProviderName %>"
40  SelectCommand="select * from policy natural join property">
41  </asp:SqlDataSource>
```

And here is the result on the webpage:

	ID	DESCRIPTION	PRICE	RENEWAL DURATION	PROPERTY CODE	AGE OF PROPERTY	PROPERTY VALUE
Select	1	esse assumenda quas in a molestiae est voluptas reiciendis.	524	2	5	18	242486
Select	2	repellendus necessitatibus saepe aut et reiciendis proident.	1267	15	1	27	642143
Select	3	nostrud eos officia consequat harum excepteur.	1155	14	1	12	654036
Select	4	saepe tenetur omnis et.	1200	28	1	15	877958
Select	5	optio dolore saepe.	3109	11	1	2	935319
Select	6	ut harum culpa laborum quod do corrupti nisi deserunt aliquip.	234	3	4	2	989204
Select	7		1445	27	4	21	776312
Select	8	sint qui id qui quas est dolore aut sint consequat. dolorum.	2042	25	4	1	254157
Select	9	mollitia ea labore et mollit delectus ex voluptates officia a. odio amet eu et fugiat.	1533	9	4	27	603511
Select	10	consequat hic et est et.	193	18	3	4	425916

1 2

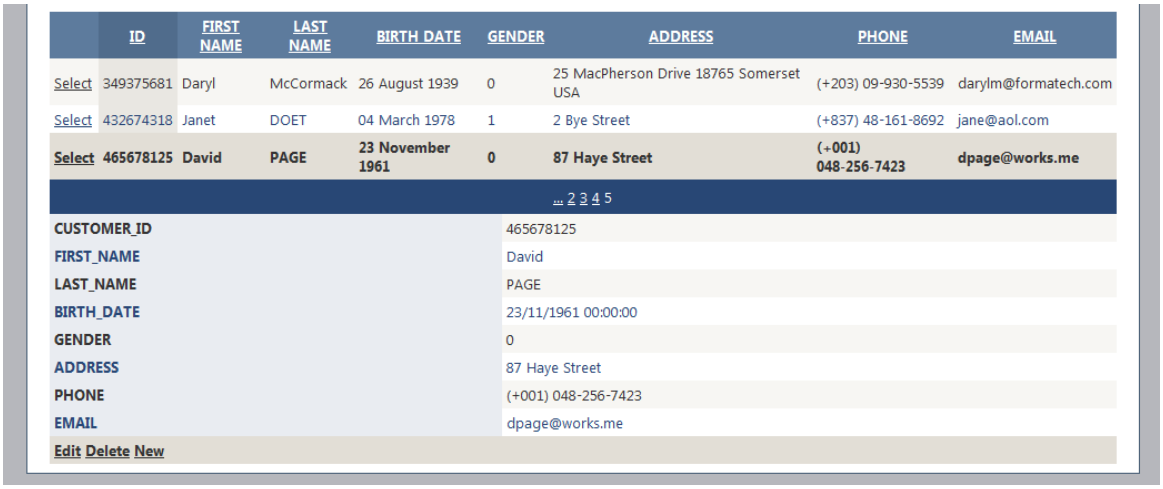
2. INSERT

On the page ~/Customers.aspx, we use the DetailsView with the Insert link/button to insert a new customer:

CUSTOMER_ID	465678125
FIRST_NAME	David
LAST_NAME	Page
BIRTH_DATE	23/11/1961
GENDER	0
ADDRESS	87 Haye Street
PHONE	(+001) 048-256-7423
EMAIL	dpage@works.me

Insert Cancel

Then we press the Insert link/button and see the result in the table (the new customer is selected in the screen capture):



	ID	FIRST NAME	LAST NAME	BIRTH DATE	GENDER	ADDRESS	PHONE	EMAIL
Select	349375681	Daryl	McCormack	26 August 1939	0	25 MacPherson Drive 18765 Somerset USA	(+203) 09-930-5539	darylm@formatech.com
Select	432674318	Janet	DOET	04 March 1978	1	2 Bye Street	(+837) 48-161-8692	jane@aol.com
Select	**465678125**	**David**	**PAGE**	**23 November 1961**	**0**	**87 Haye Street**	**(+001) 048-256-7423**	**dpage@works.me**

... 2 3 4 5

CUSTOMER_ID	465678125
FIRST_NAME	David
LAST_NAME	PAGE
BIRTH_DATE	23/11/1961 00:00:00
GENDER	0
ADDRESS	87 Haye Street
PHONE	(+001) 048-256-7423
EMAIL	dpage@works.me

Edit Delete New

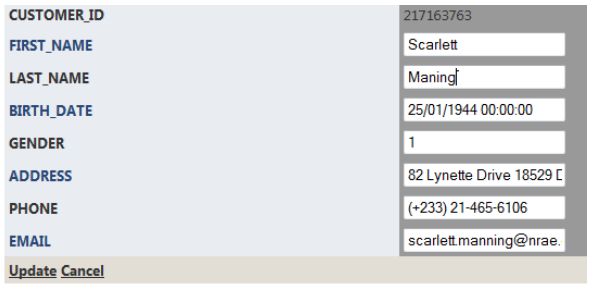
As expected.

3. UPDATE

Again on the page ~/Customers.aspx, we select customer ID 217163763:

	ID	FIRST NAME	LAST NAME	BIRTH DATE	GENDER	ADDRESS	PHONE	EMAIL
Select	**217163763**	**Scarlett**	**Manning**	**25 January 1944**	**1**	**82 Lynette Drive 18529 Denver USA**	**(+233) 21-465-6106**	**scarlett.manning@nrae.com**

Then we use the DetailsView with the Edit link/button to update Scarlett's last name to Maning (in a real world scenario, perhaps this had been a typo that we are now fixing):



CUSTOMER_ID	217163763
FIRST_NAME	Scarlett
LAST_NAME	Maning
BIRTH_DATE	25/01/1944 00:00:00
GENDER	1
ADDRESS	82 Lynette Drive 18529 D
PHONE	(+233) 21-465-6106
EMAIL	scarlett.manning@nrae.

Update Cancel

Then we click update and look at the updated record in the table:

	ID	FIRST NAME	LAST NAME	BIRTH DATE	GENDER	ADDRESS	PHONE	EMAIL
Select	**217163763**	**Scarlett**	**MANING**	**25 January 1944**	**1**	**82 Lynette Drive 18529 Denver USA**	**(+233) 21-465-6106**	**scarlett.manning@nrae.com**

Ah, you must have noticed that Scarlett's last name has turned to uppercase. This is because of a trigger that we defined in Level 2, which turns the last name of customers on insert or update to uppercase. This is therefore very much expected.

4. DELETE

This time, on the page ~/Claims.aspx, we arbitrarily delete claim ID 4, which was applied on policy ID 1579, as we see on the picture below:

	CLAIM ID	CUSTOMERS POLICY ID	CLAIM DATE	DESCRIPTION	AMOUNT	STATUS	RESULT	SETTLEMENT DATE
Select	2	463	27/05/1993 00:00:00	minus dolore asperiores nostrud blanditiis impedit corrupti asperiores tempor enim. itaque nobis.	94842	3	eveniet qui rerum do esse excepteur voluptate assumenda voluptates incididunt.	06/11/1991 00:00:00
Select	3	993	12/08/1999 00:00:00	adipisicing aut labore magna facilis exercitation.	40466	4	irure eiusmod corrupti enim fugiat officiis ut aut est excepteur. possimus deserunt.	02/09/1984 00:00:00
Select	4	1579	09/04/1993 00:00:00	dolores tempor est minim pariatur soluta cupiditate adipisicing voluptates distinctio. est qui et.	96169	8	et eu in quis sint autem sint officia non.	21/04/1991 00:00:00

Let us select it and press the DELETE link/button:

CLAIM_ID	4
CUSTOMERS_POLICY_ID	1579
CLAIM_DATE	09/04/1993 00:00:00
DESCRIPTION	dolores tempor est minim pariatur soluta cupiditate adipisicing voluptates distinctio. est qui et.
AMOUNT	96169
STATUS	8
RESULT	et eu in quis sint autem sint officia non.
SETTLEMENT_DATE	21/04/1991 00:00:00
Edit Delete New	

And it now nowhere to be found: when ordered by ascending ID, we see the table going from ID 2 to 3:

	CLAIM ID	CUSTOMERS POLICY ID	CLAIM DATE	DESCRIPTION	AMOUNT	STATUS	RESULT	SETTLEMENT DATE
Select	**2**	**463**	**27/05/1993 00:00:00**	**minus dolore asperiores nostrud blanditiis impedit corrupti asperiores tempor enim. itaque nobis.**	**94842**	**3**	**eveniet qui rerum do esse excepteur voluptate assumenda voluptates incididunt.**	**06/11/1991 00:00:00**
Select	3	993	12/08/1999 00:00:00	adipisicing aut labore magna facilis exercitation.	40466	4	irure eiusmod corrupti enim fugiat officiis ut aut est excepteur. possimus deserunt.	02/09/1984 00:00:00
Select	5	1230	14/05/1987 00:00:00	iusto dolor eu.	36762	7	quidem ipsum a in esse ut ut voluptas et deserunt. eveniet omnis aut iusto.	25/01/1990 00:00:00

5. PROCEDURE

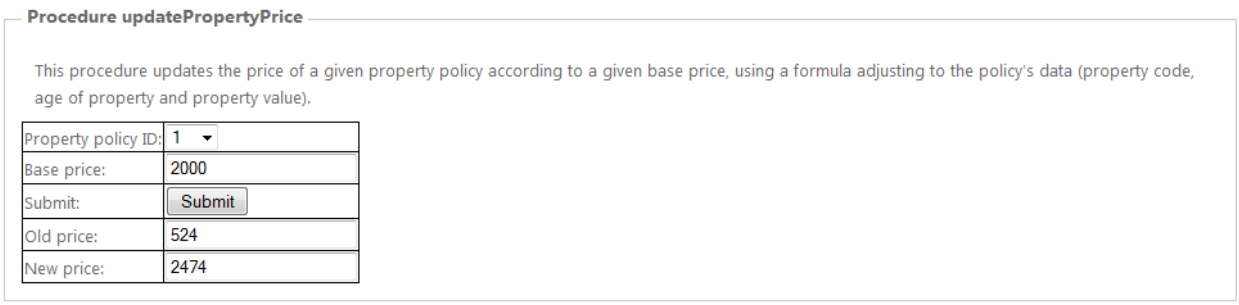
On the page ~/ProcFunc.aspx, we go to the “Procedure updatePropertyPrice” panel we arbitrarily select property policy ID 1 and enter 1000 as the base price for the new price of the selected policy.

Procedure updatePropertyPrice

This procedure updates the price of a given property policy according to a given base price, using a formula adjusting to the policy's data (property code, age of property and property value).

Property policy ID:	1
Base price:	2000
Submit:	Submit
Old price:	
New price:	

We now just have to press Submit and see the result:



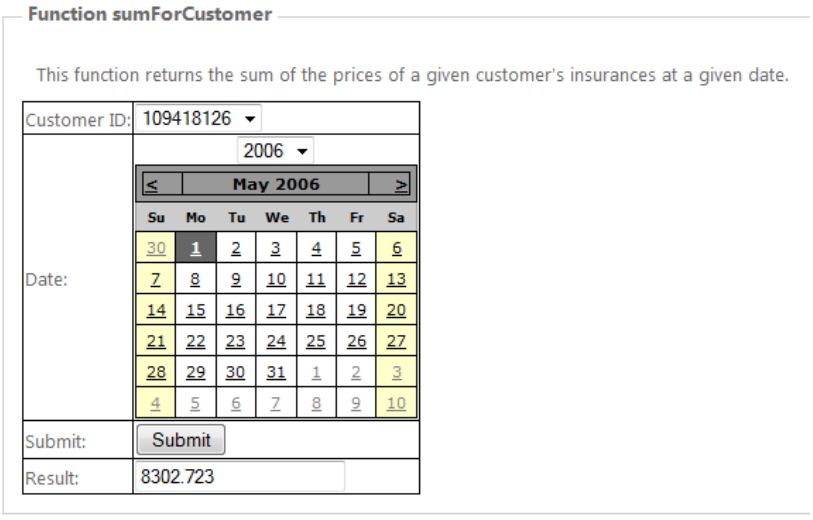
We can check in ~/PropertyPolicies.aspx that this is indeed true: below are the comparisons of before/after the procedure execution:

ID		DESCRIPTION	PRICE	RENEWAL DURATION	PROPERTY CODE	AGE OF PROPERTY	PROPERTY VALUE
Select	1	esse assumenda quas in a molestiae est voluptas reiciendis.	574	2	5	18	242486

ID		DESCRIPTION	PRICE	RENEWAL DURATION	PROPERTY CODE	AGE OF PROPERTY	PROPERTY VALUE
Select	1	esse assumenda quas in a molestiae est voluptas reiciendis.	2474	2	5	18	242486

6. FUNCTION

Finally, again on the page ~/ProcFunc.aspx, we go to the "Function sumForCustomer" panel we choose to get the sum of policy prices for the customer ID 109418126, then on the DropdownList we select year 2006, the Calendar automatically moves to year 2006 (we implemented this because pressing 7 times the back arrow on the Calendar can be tiring), then we select May 1st, and finally we submit:



We have already proven the accuracy of this function in Level 2 and despite the fact that the policy prices have been modified since then, the function has not, and remains valid. We can therefore trust this result.

4. Conclusion

In retrospect, this project gave us the occasion to look at a lot of applications, methods and concepts:

- Model conception of RDBMS using DDS-Lite v2.21 with entities, relationships attributes, keys, normal form.
- SQL queries on an Oracle 11g express database with many types of operators.
- Writing views, indexes, commit/rollback, constraints, triggers, PL procedures, PL functions, PL programs and reports with substitution variables on PL/SQL developer v10.
- Connecting an Oracle database to a Java program from Netbeans or a C# program from Visual Studio 2010, as well as developing an ASP.NET GUI based on C# in a few days from very little knowledge.

On a personal note, this was also the first truly organized project for which we used a Control Versioning System (in this case, git, with a Github remote repository). At the time of writing, it is publically available on https://github.com/fedidat/Mini-project_in_Databases. All code written by us is licensed under the GPLv2.

We would like to thank Aryeh Wiesen for helping us along the way on the few occasions during which we struggled.