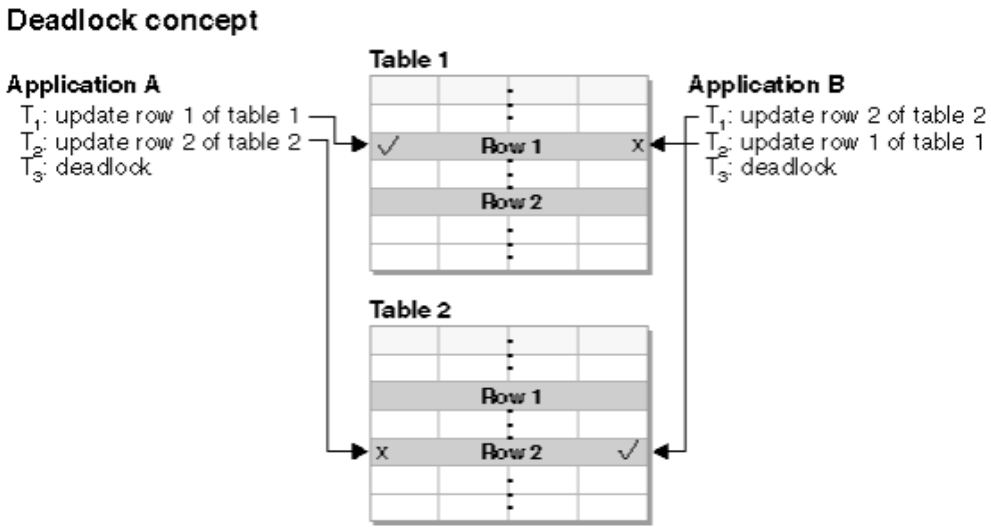
ZADACI ZA DATABASE ADMINISTRATORA

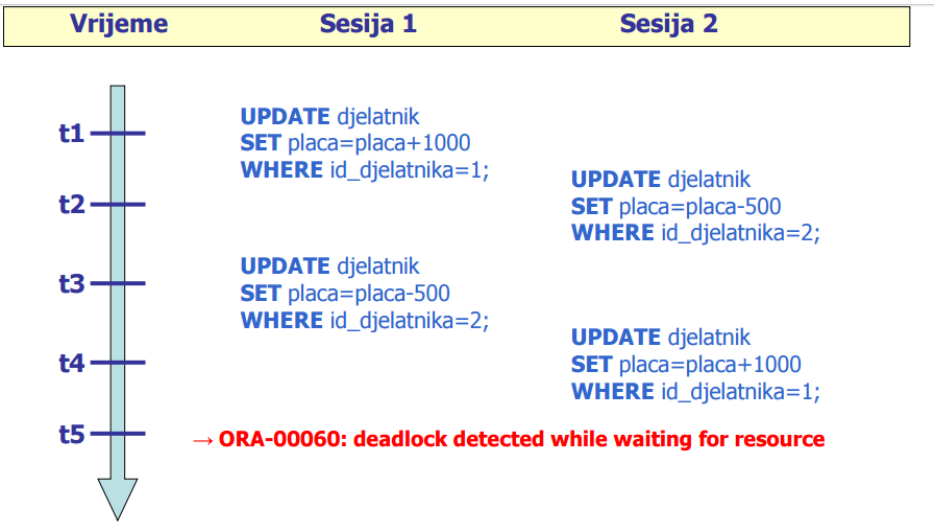
1. Potrebno je na proizvoljan način simulirati deadlock na bazi podataka. Opisati potrebne korake te što je i kako dovelo do deadlocka između 2 procesa. Objasniti što treba promijeniti kako se taj specifični deadlock ne bi ponovio.

Deadlock - Situacija u kojoj dva (ili više) procesa čekaju jedan na drugoga.

Mrtva točka (deadlock) nastaje kada dvije (ili više) transakcija, obje (sve) čekaju da se otpusti zaključavanje koje drži druga transakcija.

Zastoj = situacija u kojoj dva (ili više) procesa čekaju jedan na drugoga, procesi se sami ne mogu osloboditi zastoja, dešava se kada proces ima isključivo pravo na neki objekt.





Transakcije prelaze u stanje čekanja. Ova pojava se naziva deadlock. Dvije transakcije svojim zahtjevima blokiraju jedna drugu, te onemogućavaju pristup podacima.
Sustavi baza podataka imaju ugrađene mehanizme za detekciju deadlock-a. Uobičajeni način razrješavanja ovakve situacije je prekid jedne od transakcija, što drugoj omogućava normalan nastavak i izvršavanje do kraja. Razumljivo deadlock nije zadovoljavajuće rješenje međudjelovanja transakcija. Umjesto toga interni mehanizmi baze podataka vrše prilagođavanje lokota strukturi same transakcije.

--cancel thread(13245ID)

2. Izvršiti skriptu:

```
IF OBJECT_ID('sales.TestProc1', 'P') IS NOT NULL
BEGIN
   EXEC('DROP PROCEDURE sales.TestProc1');
END;
GO
CREATE PROCEDURE Sales.TestProc1
@ProductID INT
AS
SELECT SalesOrderDetailID, OrderQty
FROM Sales.SalesOrderDetail
WHERE ProductID = @ProductID;
GO

EXEC Sales.TestProc1 @ProductID=870
GO

EXEC Sales.TestProc1 @ProductID=897
GO

SELECT SalesOrderDetailID, OrderQty
FROM Sales.SalesOrderDetail
WHERE ProductID = 870;
GO

SELECT SalesOrderDetailID, OrderQty
FROM Sales.SalesOrderDetail
WHERE ProductID = 897;
GO
```

Objasniti razliku u izvođenjima upita (da li neki traju kraće/dulje, zahtijevaju više resursa itd.). Preporučiti optimizaciju procedure ako je potrebna.

Izvođenje upita – brzo izvršavanje.

SQLQuery18.sql - D...8P06UK\Ivana (57))* | Object Explorer

```
IF OBJECT_ID('sales.TestProc1', 'P') IS NOT NULL
BEGIN
    EXEC('DROP PROCEDURE sales.TestProc1');
END;
GO
CREATE PROCEDURE Sales.TestProc1
@ProductID INT
AS
SELECT SalesOrderDetailID, OrderQty
FROM Sales.SalesOrderDetail
WHERE ProductID = @ProductID;
GO

EXEC Sales.TestProc1 @ProductID=870
GO
```

100 %

Results | Messages

	SalesOrderDetailID	OrderQty
10	36261	18
11	36305	5
12	36407	3
13	36423	6
14	36452	10
15	36521	9

Results | Messages

```
(4688 row(s) affected)

(2 row(s) affected)

(4688 row(s) affected)

(2 row(s) affected)
```

3. Izvršiti skriptu:

```
IF OBJECT_ID('sales.TestProc2', 'P') IS NOT NULL
BEGIN
   EXEC('DROP PROCEDURE sales.TestProc2');
END;
GO

CREATE PROCEDURE [Sales].TestProc2
   @yearFrom SMALLINT
 , @yearTo SMALLINT
 AS
DECLARE @year SMALLINT;

DECLARE c CURSOR
FOR
SELECT DISTINCT YEAR([OrderDate]) OrderDate FROM [Sales].[SalesOrderHeader] ORDER BY
OrderDate;
OPEN [c];

DECLARE @result TABLE (
    OrderYear INT
  , OrderMonth INT
  , [TotalOrderQty] INT
  , [TotalPrice] MONEY
   );


FETCH NEXT FROM [c] INTO @year;
WHILE @@FETCH_STATUS = 0
BEGIN
INSERT INTO @result([OrderYear], [OrderMonth], [TotalOrderQty], [TotalPrice])
SELECT
[t].[OrderYear], [t].[OrderMonth], SUM([t].[OrderQty]) [TotalOrderQty], SUM([t].[OrderQty] *
([t].[UnitPrice] - [t].[UnitPriceDiscount])) [TotalPrice]
FROM (SELECT YEAR([soh].[OrderDate]) OrderYear, MONTH([soh].[OrderDate]) OrderMonth,
[sod].[OrderQty], [sod].[UnitPrice], [sod].[UnitPriceDiscount]
FROM [Sales].[SalesOrderDetail] sod JOIN [Sales].[SalesOrderHeader] soh ON
[soh].[SalesOrderID] = [sod].[SalesOrderID] ) t
WHERE [t].[OrderYear] = @year
GROUP BY [t].[OrderYear], [t].[OrderMonth]
ORDER BY [t].[OrderYear], [t].[OrderMonth];
FETCH NEXT FROM [c] INTO @year;
END;
CLOSE [c];
DEALLOCATE c;
SELECT * FROM @result;
GO
```

Optimizirati gore navedenu proceduru – promijeniti kod gdje i ako je potrebno, dodati objekte u bazi koji bi ubrzali izvođenje.

Izvršavanje je bilo brzo.

SQLQuery5.sql - DE...8P06UK\Ivana (53)) | SQLQuery3.sql - DE...8P

```
USE [AdventureWorks2014]
GO

DECLARE @RC int
DECLARE @yearFrom smallint
DECLARE @yearTo smallint

-- TODO: Set parameter values here.

EXECUTE @RC = [Sales].[TestProc2]
   @yearFrom
  ,@yearTo
GO
```

100 %

Results | Messages

	OrderYear	OrderMonth	TotalOrderQty	TotalPrice
1	2011	9	157	502073,8458
2	2011	12	1040	1309976,5047
3	2011	6	141	458910,8248
4	2011	7	2209	2046259,916
5	2011	10	5382	4590869,0987
6	2011	5	825	503805,9169
7	2011	11	230	737839,8214
8	2011	8	2904	2496360,7024
9	2012	9	8294	3455326,7945
10	2012	3	3184	2976349,4071

100 %

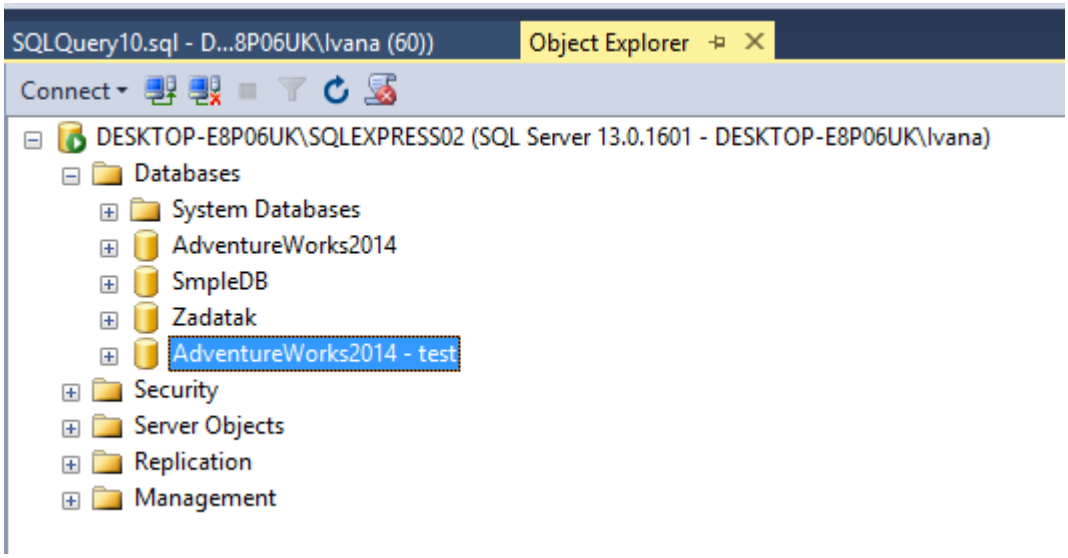
Results | Messages

```
(8 row(s) affected)

(12 row(s) affected)

(12 row(s) affected)

(6 row(s) affected)

(38 row(s) affected)
```

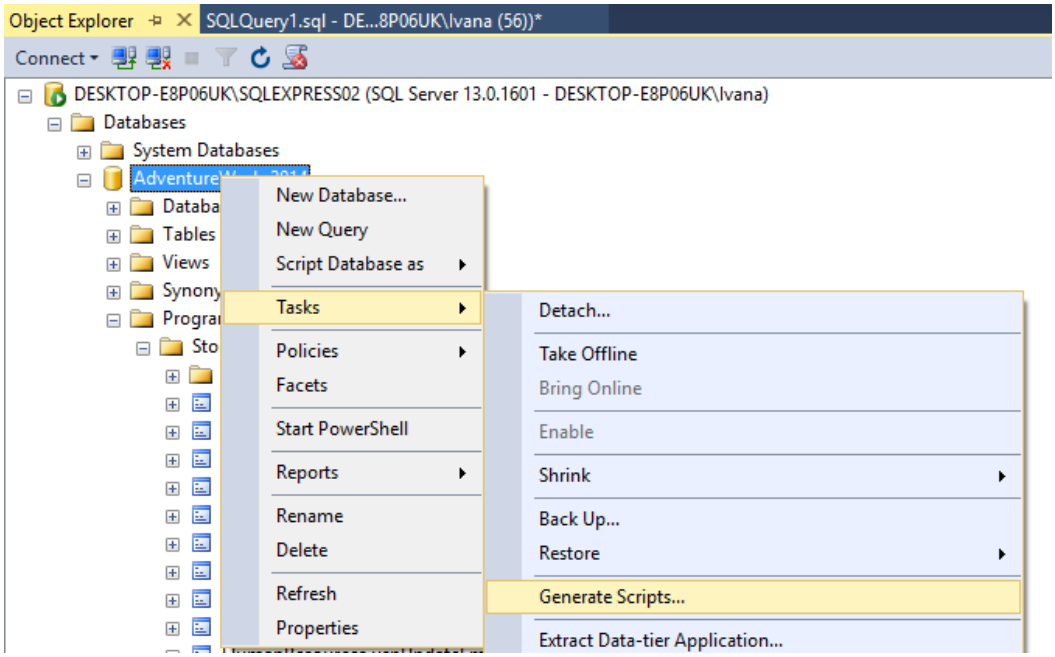
4. Napisati proceduru koja vraća zadnjih 5 upita koji su timeoutali, njihovo trajanje prije timeouta i podatke o pozivatelju (naziv stroja s kojeg je došao poziv, IP adresa, login, i sl.)

5. Za potrebe testova potrebno je na dnevnoj bazi kreirati bazu koja je replika produkcijske baze. Napisati skriptu koja rekreira testnu bazu iz produkcijske – sadrži sve objekte, podatke, usere itd. koji postoje u produkcijskoj bazi. Koristiti proizvoljnu metodu/alat za kreiranje kopije.

Možemo i iskreirati skripta.sql



Možemo i iskreirati skripta.sql



Napomene:
Za sve zadatke **koristiti AdventureWorks2014 bazu**. Dopuštene su modifikacije baze uz dokumentiranje promjene u pripadnom zadatku.

Skripte neka budu za compatibility mode: SQL Server 2014 Standard Edition.

Za sva pitanja slobodno se javiti na: andrej.mihajlovic@infobip.com