UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Urban Zupančič

Uporaba SQL Service Broker-ja za integracijo dveh informacijskih sistemov

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: dr. Damjan Vavpotič

Ljubljana 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00125/2011
Datum: 06.04.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **URBAN ZUPANČIČ**

Naslov: **UPORABA SQL SERVICE BROKERJA ZA INTEGRACIJO DVEH INFORMACIJSKIH SISTEMOV**

USE OF SQL SERVICE BROKER FOR INTEGRATION OF TWO INFORMATION SYSTEMS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Preučite in predstavite uporabo orodja SQL Service Broker za povezavo dveh informacijskih sistemov. Primerjajte uporabo SQL Service Brokerja in enostavnejše alternativne rešitve z uporabo tabel in pogledov, ki se prav tako lahko uporabi za povezavo dveh informacijskih sistemov. V nalogi se posvetite tako praktičnim kot teoretičnim vidikom obeh pristopov ter predstavite njihove prednosti in slabosti. Predstavite tudi uporabo SQL Service Brokerjav na praktičnem primeru.

Mentor:

viš. pred. dr. Damjan Vavpotič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Urban Zupančič,

z vpisno številko 63040183,

sem avtor diplomskega dela z naslovom:

Uporaba SQL Service Broker-ja za integracijo dveh informacijskih sistemov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom dr. Damjana Vavpotiča
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 14.10.2011 Podpis avtorja:

Zahvala

Zahvaljujem se vsem ki so mi pomagali pri izdelavi diplomske naloge, še posebej mentorju dr. Damjanu Vavpotiču.

Posebej se zahvaljujem staršem, ki so me motivirali, financirali ter verjeli vame tekom študija.

Kazalo

1 UVOD 3

2 SERVICE BROKER 4

2.1 O SERVICE BROKERJU 4
2.1.1 Minimalne zahteve SB-ja 4
2.1.2 Predstavitev SB 4
2.1.3 Delovanje SB-ja 4
2.1.4 Tipična uporaba Service Broker-ja 6
2.1.5 Prednosti in slabosti SB-ja 7
2.1.6 Alternative SB-ju 7
2.1.7 Prejemanje sporočila 8
2.1.8 Obvladovanje napak 8
2.2 PRIMERJAVA SB-JA IN VMESNE TABELE 10
2.2.1 Kreacija vseh tabel 10
2.2.2 Vmesna tabela 11
2.2.3 Kreacija SB procedure 16
2.3 UPORABA SB–JA NA PRIMERU V PRAKSI 20
2.3.1 Predstavitev razlogov za uporabo SB-ja v praksi 20
2.3.2 Zakaj pravzaprav sploh potrebujemo SB? 21
2.3.3 Struktura sistema ali okolja? 22
2.3.4 Primer XML Dokumenta dobljenega sporočila iz IPS-a in obdelava le-tega 25
2.3.5 Primer dobljenega poročila poslanega iz MePis-a v IPS 27
2.3.6 Potek sporočila iz enega v drug sistem preko SB-ja 29
2.3.7 Zaključevanje dokumenta in pošiljanje sporočila pošiljatelju 32
2.3.8 Pretvorba zapisa iz tabele v XML dokument 34
2.3.9 Mogoči problemi, do katerih lahko pride. 35

3 SKLEPNE UGOTOVITVE 36

4 PRILOGE 37

5 KAZALO SLIK 38

6 VIRI 39

Seznam uporabljenih kratic

SB	Service Broker
ERP	Enterprise Resource Planning
XML	Extensible Markup Language
SQL	Structured Query Language
MSMQ	Microsoft Message Queue
SOAP	Simple Object Access Protocol
WSDL	Web Service Definition Language
WCF	Windows Communication Foundation
MS	Microsoft

Povzetek

Diplomsko delo opisuje uporabo Microsoftovega Service Broker-ja (SB). To orodje dobimo kot del Microsoftovega SQL Serverja (od verzije 2005 dalje) in se uporablja za prenos podatkov med dvema instancama PB na lokalnem omrežju, med samim seboj ali z bazo na oddaljenem strežniku. Opisal bom delovanje SB-ja, njegovo alternativo ter opisal uporabo te tehnologije v projektu, v katerem sem SB uporabil.

Kot alternativo SB-ju sem izbral vmesno tabelo, narejeno v SQL Serverju. Tehnologija je podobna SB-ju. Ta tehnologija je del programov SQL Serverja. Prednost pri uporabi vmesne tabele je ta, da je zelo enostavno narediti tak sistem, vendar to ne nagne tehtnice na stran vmesnih tabel.

Za konec bom s primeri predstavil delovanje SB-ja v naši informacijski rešitvi. Na projektu smo bili primorani uporabiti SB zato, ker imajo v hrvaškem podjetju JGL ERP sistem, ki ga ni razvilo naše podjetje. Za prenos podatkov med dvema sistemoma smo se odločali med dvema tehnologijama in sicer med SB-jem in uporabo vmesne tabele. Na koncu smo izbrali SB, saj je bil zelo močan argument njegova zanesljivost.

Abstract

The focus of the diploma thesis is set on Microsoft SQL Servis Broker (SB). Service Broker is part of the Microsoft Sql Server(from version 2005) and is used for data transfer between two data bases on local or remote networks or between itself.

I will describe operation of SB, his alternative and usage of this tool in my project, where I used this tool. As an alternative of SB I used intermediate tables, made in SQL Server. This tool is similar to SB.

For end I will demonstrate usage of SB in our IT solution. We used SB because of the gap between two databases mad by different IT enterprises.

1 Uvod

Veliko srednjih in tudi velikih podjetij postopoma informatizira svojo proizvodnjo. Do tega pride zaradi omejevanja sredstev ali pa zaradi namernega počasnega vpeljevanja informatike v proizvodni proces. S tem se omili tveganje za prekinitev celotne proizvodnje ter razporedi financiranje na daljše obdobje. Pri postopni informatizaciji se veliko podjetij odloči, da različna podjetja z različnimi programi informatizirajo različne dele podjetja. Zaradi različnih programov in tako tudi različnih podatkovnih baz, nastanejo veliki prepadi med bazami. Baze morajo biti ves čas med seboj povezane, saj morajo biti podatki konsistentni med podatkovnimi bazami. Med bazami je potrebno vzpostaviti takšno povezavo, da se bodo lahko podatki izmenjevali. Dve izmed teh tehnologij sta tudi vmesne tabele med bazami ter naprednejši SB.

Za povezavo med bazama smo uporabili SB, preko katerega se prenašajo podatki. Ta povezava skrbi, da sta bazi vedno ažurni ter konsistentni med podatkovnimi bazami.

V okviru praktičnega dela naloge sem vzpostavil povezavo med dvema podatkovnima bazama s pomočjo SB-ja in vmesne tabele. Oba pristopa sem med seboj primerjal in na koncu ugotovil, da je SB v vseh primerih boljši od vmesnih tabel.

2 Service Broker

2.1 O Service Brokerju

2.1.1 Minimalne zahteve SB-ja

SB je del produkta Microsoftovega SQL Serverja. Vsebuje ga vsaka instanca SQL Server-ja. Zahteve SB-ja so torej kar zahteve SQL Serverja. Minimalne zahteve za nastavitev SQL Serverja 2008 R2 (Standard Edition) so:

- *Intel pentium III 600 MHz* ali *hitrejši*,
- *512 MB RAM-a* ali *več*,
- *vsaj 750 MB prostora na disku* za nastavitev *podatkovnega strežnika,*
- *potrebujemo vsaj Windows Xp* ali *novejšo različico*.

2.1.2 Predstavitev SB

SB je transportni mehanizem med eno, dvema ali več bazami. S SB-jem lahko notranji ali zunanji procesi pošiljajo asinhrona sporočila, za katera imamo dokazila, da jih je naslovnik dobil oz. ni dobil. Sintaksa je podobna sintaksi v SQL procedurah, vendar z nekaj dopolnili. SB lahko uporabljamo za prenos sporočil oziroma podatkov med podatkovnimi bazami. Bazi se lahko fizično nahajata na istem strežniku, lahko sta na dveh različnih, ki sta v lokalni mreži ali pa sta povezani preko interneta.

Terminologija SB-ja:

- **Sporočilo** *(Message)* – informacija, ki se izmenja med dvema aplikacijama, ki uporabljata SB. Sporočilo se lahko preveri, da se ujema z določeno XML shemo.
- **Pogovor** *(conversation)* – je zanesljivo, dolgotrajno in asinhrono izmenjevanje sporočil.
- **Dialog** (*dialog*) – dialog med dvema servisoma. Vsi pogovori med SB-jem so dialogi.
- **Pošiljatelj** *(initiator)* – udeleženec, ki začne dialog.
- **Prejemnik** *(target)* – udeleženec, ki sprejme dialog, ki ga je začel pošiljatelj.
- **Pogovorna grupa** *(conversation group)* – skupina povezanih pogovorov. Vsak pogovor pripada samo določeni grupi.
- **Pogodba** *(contract)* – dogovor med dvema servisoma, ki nam pove, kateri tipi sporočil so dovoljeni v pogovoru.
- **Vrsta** *(queue)* – v njej so shranjena sporočila določenega servisa.
- **Servis** *(service)* – določeno opravilo, ki lahko pošilja in sprejema sporočila.

2.1.3 Delovanje SB-ja

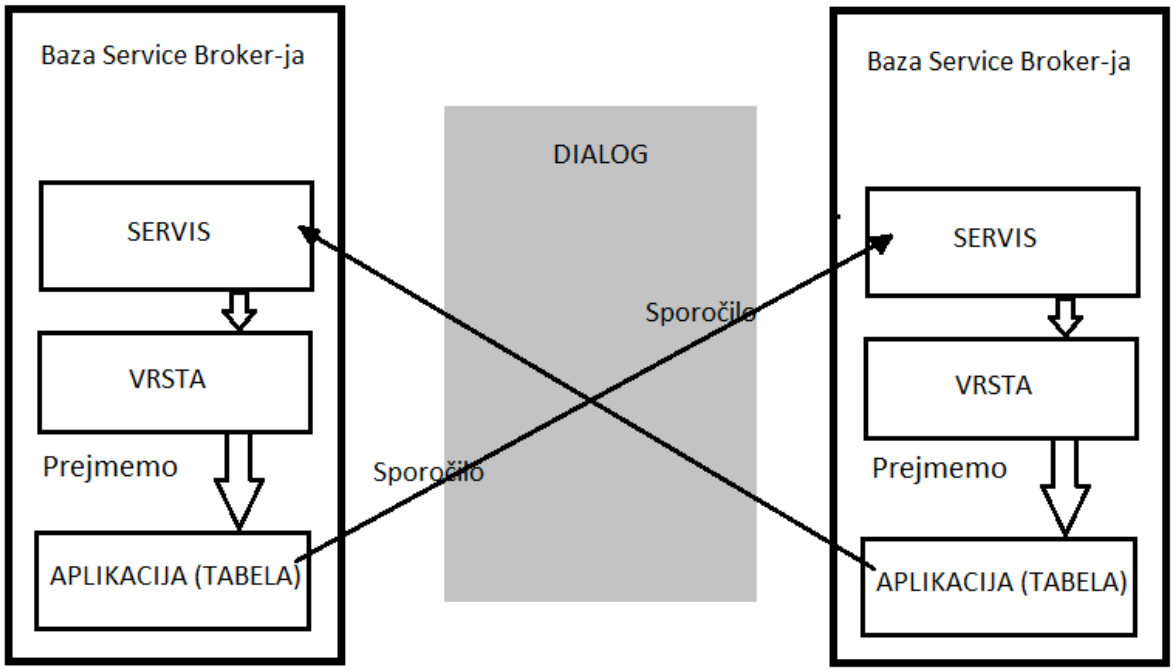
Sporočila, ki jih SB pošilja, se shranjujejo v pošiljateljevi bazi: pošljemo jih lahko kadar koli. Ena izmed možnosti pošiljanja je ta, da se lahko sporočila pošljejo ob točno določenem času. Za uporabo te lastnosti uporablja SB vrste (Slika 1). V vrstah se shranjujejo sporočila, ki se pošiljajo v pošiljateljevo bazo. Vrste se uporabljajo za ohlapne vezi med pošiljateljem in prejemnikom. Ko pošiljatelj pusti sporočilo v vrsti, je naloga SB-ja, da doseže prejemnika. Sporočila se lahko pošiljajo točno določenemu prejemniku.

Dialog med pošiljateljem in prejemnikom je dvosmerna komunikacija. Sporočila so dostavljena v enakem vrstnem redu, kot so bila poslana. S tem se zagotavlja, da so vsa sporočila dostavljena. Če sporočilo ni dostavljeno zaradi prekinitve ali kakšne druge napake, se sporočilo shrani in se pošlje takoj, ko je prenos mogoč. Identifikacija teh sporočil je določena s tako imenovanimi »conversation handle-i«, ki so unikatni identifikatorji. Le-ti so povezani z pogovornimi okni in tako točno vemo, h kateremu oknu pripada sporočilo. S to funkcionalnostjo lahko pošiljamo več sporočil hkrati, ne da bi se sporočila pomešala.

Sporočila se združujejo v grupe. Ena grupa je eno pogovorno okno oziroma ena aplikacija. Ko je prebrano prvo sporočilo grupe, se le-ta zaklene s strani niti prejemnika in se ji nadene identifikator grupe. Samo nit prejemnikove transakcije lahko to sporočilo odklene. Z grupami se izognemo temu, da bi eno sporočilo obdelovalo več niti.

SB se aktivira s proceduro, ki jo sami določimo. Ko sporočila pridejo do storitve, SB preveri ali procedura za omenjeno sporočilo že teče. Če še ne, jo zažene. Procedura izvaja sporočila tako dolgo, dokler vrsta ni prazna. Če sporočila prihajajo v vrsto hitreje, kot jih lahko procedura procesira, SB zažene nov primer procedure. Zažene jih lahko toliko, da se le-ta v vrsti ne nabirajo. Prav tako se ne zapisujejo sproti v tabelo, temveč čakajo, da so vsa uspešno prenesena na prejemnikov cilj. Če je transakcija neuspešna, se vsa sporočila zbrišejo. Pošiljatelju se pošlje ukaz, da je potrebno ponovno poslati vsa sporočila. Transakcija se potrdi samo takrat, ko so vsa sporočila obdelana.

Dobra stvar SB-ja je ta, da so tako sporočila kot tudi podatki v eni bazi. Tako ne more priti do izgube podatkov. To se zna zgoditi po restavriranju baze, saj lahko pride do ne-sinhroniziranih podatkov med bazo in sporočili. Če uporabljamo prenos sporočil med dvema bazama na enem SQL Serverju, lahko sporočila neposredno damo v prejemnikovo vrsto, kjer čakajo na obdelavo. S tem povečamo performanco SB-ja, saj se izognemo sporočilom v pošiljateljevi vrsti.



Slika 1: Pošiljanje sporočil

2.1.4 Tipična uporaba Service Broker-ja

Tehnologija SB-ja je zelo koristna pri procesih, ki uporabljajo oziroma potrebujejo asinhrono procesiranje podatkov ter pri procesih, v katerih je potrebno poslati podatke na več različnih računalnikov.

Tipične uporabe SB so [4]**:**

- **Asinhroni sprožilci** *(Asynchronous Triggers),*
- **Zanesljivo procesiranje vrst** *(Reliable Query Processing),*
- **Zanesljivo zbiranje podatkov** *(Reliable Data Collection),*
- **Distribucija podatkov med več podatkovnimi bazami** *(Distributed Server-Side Processing for Client Applications),*
- **Združevanje podatkov za aplikacijo** *(Data Consolidation for Client Applications).*

2.1.4.1 Asinhroni sprožilci (Asynchronous Triggers)

SB lahko uporabljamo namesto sprožilcev. Ko se sproži potrditev originalne procedure, SB dostavi sporočilo do cilja. Ker se to izvaja ločeno od originalne procedure, se lahko napravi potrditev. Zaradi ločenega delovanja se izognemo možnemu upočasnjevanju sistema. Pri sprožilcu ostane originalna procedura odprta toliko časa, kolikor je potrebno, da se izvede.

2.1.4.2 Zanesljivo procesiranje poizvedb *(Reliable Query Processing)*

Nekatere aplikacije potrebujejo zanesljiv prikaz poizvedb. V računalništvu lahko pride do različnih prekinitev sistema, kot so okvara računalnika, nedelovanje mreže oziroma interneta ali izpad električne energije. S SB-jem se lahko tem prekinitvam izognemo, če pošljemo sporočila v vrsto. Aplikacija bo obdelala vsa sporočila v enaki transakciji, ko bodo prišla na vrsto. Če bo medtem prišlo do prekinitve delovanja, se bo izvedla razveljavitev (rollback) in sporočilo se bo vrnilo nazaj v vrsto. Obdelano bo takoj, ko bo težava odpravljena.

2.1.4.3 Zanesljivo zbiranje podatkov *(Reliable Data Collection)*

Velikokrat moramo zbrati podatke iz več podatkovnih baz v eno centralno. S SB-jem je to zelo lahko. Preko njega lahko pošljemo podatke iz več pošiljateljevih mest. Ker SB zagotavlja zanesljive, asinhrone povezave, se bodo sporočila dostavila tudi takrat, ko bo prišlo do kakšne prekinitve med centralno podatkovno bazo in virom. Varnost SB-ja pomaga sporočilom, da med prenosom niso spremenjena ali napačno dostavljena.

2.1.4.4 Distribucija podatkov med več podatkovnimi bazami (Distributed Server-Side Processing for Client Applications)

Velike aplikacije, ki imajo dostop do več podatkovnih baz, si lahko ogromno pomagajo s SB-jem, saj lahko nemoteno delamo tudi takrat, ko so nekatere baze prepolne, preobremenjene ali izklopljene. Sporočila se oddajo v vrste in tam čakajo, dokler ni pravi trenutek za njihovo obdelavo. primer

2.1.4.5 Dodajanje podatkov v skupino za aplikacijo *(Data Consolidation for Client Applications)*

Aplikacije, ki morajo prikazati več informacij istočasno, pridobijo zelo veliko hitrosti s SB-jem. Za zbiranje informacij uporabimo vzporedno pridobivanje sporočil, ki so veliko hitrejša kot zaporedno zbiranje. Tudi podatki za poizvedbe se pošiljajo vzporedno. Vrnjene podatke aplikacija zbere in prikaže.

2.1.5 Prednosti in slabosti SB-ja

SLABOSTI:

- Če je prejemnik izključen, pošiljatelj vedno znova in znova pošilja sporočila v vrsto za pošiljanje. Ko prejemnik vzpostavi sistem, ga v vrsti čaka več enakih sporočil … če se hočemo temu izogniti, je potrebno dodati kontrolo v proceduro, ki preveri, ali je prejemnik vklopljen. Prihajanje do »Poison Message-ov«.
  .

PREDNOSTI:

- garancija SB-ja, da se podatki ne bodo izgubili (ali bodo prišli na cilj ali pa bodo ostali v pošiljateljevi vrsti);
- obveščanje preko mail-a, sms-a;
- enostaven prenos podatkov;
- enostavno sestavljanje xml dokumenta za prenos podatkov;
- enostavno razčlenjevanje xml dokumenta;
- uporaba nam že znane tehnologije vpisa podatkov v vrsto (vpisovanje preko vpisnih stavkov);
- centralizirana administrativna orodja – SB uporablja enaka orodja, kot jih uporablja SQL Server oz. se upravlja SB v Management Studiu.

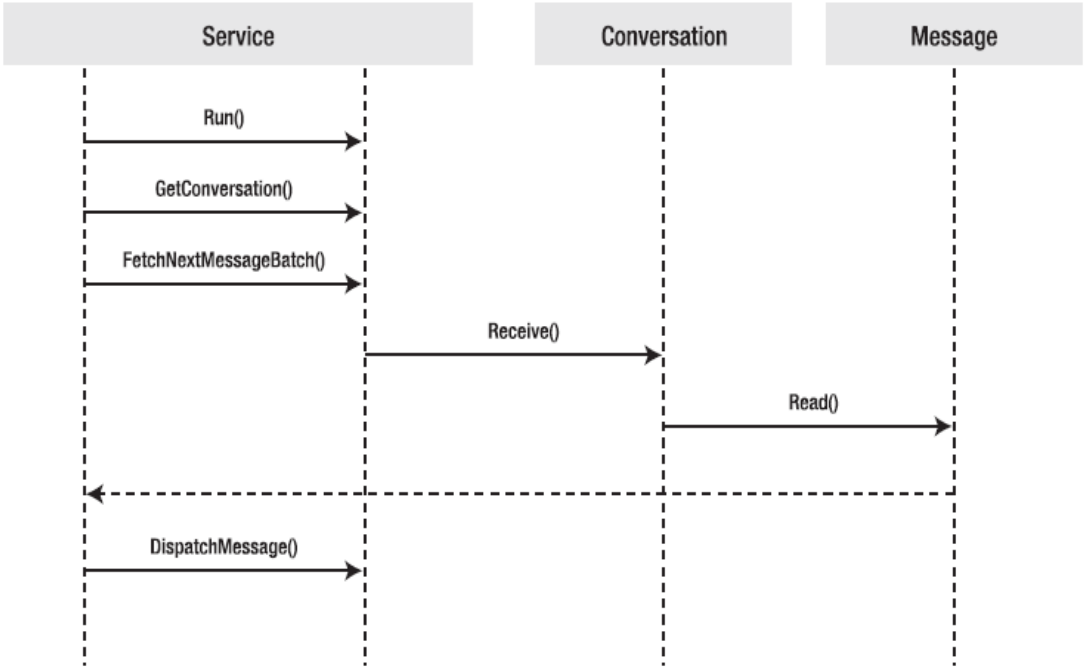
2.1.6 Alternative SB-ju

Obstaja nekaj tehnologij za izmenjavo sporočil med bazami. Na razpolago so:

- Microsoft-ov MSMQ (Microsoft Message Queue), BizTalk in Queued Components ki je del COM+ infrastrukture,
- XML Web Services temeljoč na odprtih standardih kot sta SOAP in WSDL,
- WCF standard, ki je del .Net frameworka,
- vmesna tabela narejena v Microsoft-ovem SQL Serverju ali uporaba View-a.

2.1.7 Prejemanje sporočila

V nadaljevanju si bomo pogledali, kako se prejme sporočilo iz ustrezne čakalne vrste in kako se opravi povratni klic. Da dosežemo ustrezno stanje, se mora zagnati več metod iz servisov, dialogov in razredov sporočil.



Slika 2: Pogoj sprožilca [1]

Slika (Slika 2) prikazuje UML diagram, ki se izvede, ko se prebere novo sporočilo iz vrste. Naslednje metode se kličejo med prejemanjem in obdelavo sporočila. Metoda GetConversation () se kliče znotraj Run metode. Run metoda se pokliče, kadar SQL Server pokliče shranjeno proceduro. GetConversation pokliče metodo FetchNextMessageBatch. Namen te procedure je, da zgradi T-SQL stavek za pridobivanje sporočil iz vrste. Takoj ko so podatki prepisani v SqlDataReader, se zažene metoda za branje sporočila (MessageReader). Ko vrne GetConversation metoda informacije o dialogu, se sproži metoda Recieve. Ko ta metoda pridobi sporočilo, ga razpošlje ustrezni povratni metodi, ki je glavni konstruktor Servisa.

2.1.8 Obvladovanje napak

Tipična začetniška napaka je, da programer pozabi ujeti napake, ki jih pošlje prejemnik. Za vsak pogovor, ki se zaključi, pošlje prejemnik zaključek dialoga. To vnaprej definirano sporočilo mora pošiljatelj ujeti. Glede na ujeto sporočilo, lahko pošiljatelj izpelje zaključek procedure. Vnaprej definirana sporočila so prazna sporočila.

Ujemanje takih sporočil naredimo tako, da vgradimo pogoje, ki obvladujejo različne situacije.

Poznamo tri sistemska sporočila, ki se tvorijo ob koncu pogovora:

- **sporočilo, ki se tvori ob uspešni izvršitvi pogovora**
- **sporočilo, ki se tvori ob neuspešni izvršitvi pogovora**
- **in sporočilo, ki se tvori, če je čas za pogovor potekel.**

Primer ujetja sporočila (uspešna izvršitev pogovora):

```
IF (@messagetypename='http://schemas.microsoft.com/SQL/
ServiceBroker/EndDialog')
BEGIN
-- Zaključimo pogovor

  update custom_jgl.sb_SendMessages
     set reply_from_ips = 1,
   reply_from_ips_datetime = GETDATE()
   where conversation_id = @ch

-- Naredimo še update v tabeli warehouse.document_header, da si označimo, da je prejemnik
uspešno prejel sporočilo.

update warehouse.document_header
   set transfer_date = GETDATE(),
           transfer_status_id = 1
 where id in (select document_header_id
                     from custom_jgl.sb_SendMessages
                     where conversation_id = @ch)
```

Poleg sistemskih sporočil SB pozna še notifikacije.

2.2 Primerjava SB-ja in vmesne tabele

Vmesna tabela je nadomestilo za SB. Naredi se na dokaj preprost način: najprej se moramo dogovoriti, na katerem strežniku bo vmesna tabela. Ko se določi kje bo, je potrebno dodeliti pravice uporabniku, s katerimi se bo dostopalo do tabele. Na pošiljateljevi strani je potrebno narediti proceduro, s katero se bo polnila tabela.

Na drugi strani je potrebno narediti opravilo, v katerem se bo zaganjala procedura, ki bo pregledovala, ali je v vmesni tabeli kakšen nov zapis, in ki bo nove zapise vpisovala v naše tabele. Tak način pa lahko zelo obremeni sistem, saj se mora opravilo zaganjati, če hočemo imeti zelo ažurne podatke, zelo pogosto. Če opravilo pade, je potrebno opravilo ročno zagnati.

2.2.1 Kreacija vseh tabel

```
CREATE TABLE Artikli_na_posiljatelju
(       id INT,
        sifra VARCHAR(32),
        naziv VARCHAR(255),
        cena DECIMAL(10,2),
        kolicina DECIMAL(10,3),
        barva VARCHAR(255),
        skladisce_id INT,
        teza DECIMAL(5,3),
        davek DECIMAL (5,2))
```

```
CREATE TABLE Artikli_na_prejemniku
(       id INT,
        sifra VARCHAR(32),
        naziv VARCHAR(255),
        cena DECIMAL(10,2),
        barva VARCHAR(255),
        skladisce_id INT,
        teza DECIMAL(5,3),
        zunanji_id INT,
posiljateljev_id INT)
```

```
CREATE TABLE Artikli_vmesna_tabela
(       id INT,
        sifra VARCHAR(32),
        naziv VARCHAR(255),
        cena DECIMAL(10,2),
        kolicina DECIMAL(10,3),
        barva VARCHAR(255),
        teza DECIMAL(5,3),
        skladiscna_lokacija INT)
```

2.2.2 Vmesna tabela

Na vmesni tabeli je narejen samo pogled (view), ki ga lahko uporablja prejemnik. S pogledom v proceduri bo prejemnik vpisoval zapise v svoje tabele.

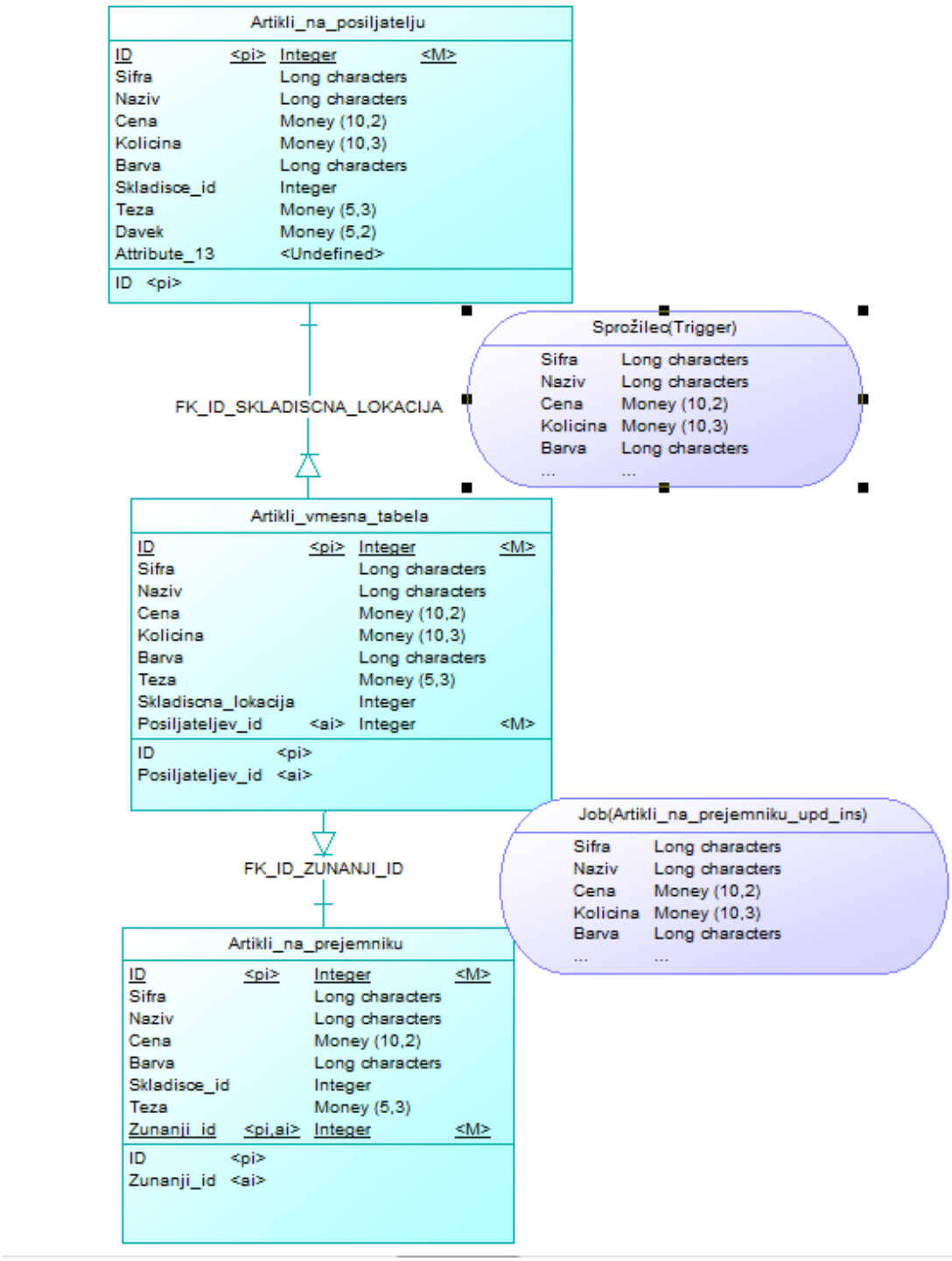
2.2.2.1 Sprožilec za vpis v vmesno tabelo

```
CREATE TRIGGER [dbo].[Artikli_na_pošiljatelju_upd_ins]
 ON [mepis_ls].[dbo].[Artikli_na_posiljatelju]
 FOR INSERT, UPDATE
 AS

 UPDATE [mepis_cc_customer].[dbo].[Artikli_vmesna_tabela]
 SET   sifra     = inserted.sifra,
          naziv  = inserted.naziv,
          cena             = inserted.cena,
          kolicina = inserted.kolicina,
          barva  = inserted.barva,
          --skladisce_id = inserted.skladisce_id          ta podatek nas ne zanima
          --teza             = inserted.teza                    ta podatek nas ne zanima
          davek  = inserted.davek
  FROM inserted
  INNER JOIN [Artikli_vmesna_tabela] AVT ON inserted.id = AVT.posiljateljev_id
```

```
INSERT INTO [mepis_cc_customer].[dbo].[Artikli_vmesna_tabela]
         (sifra,
         naziv,
         cena,
         kolicina,
         barva,
           --skladisce_id,                    ta podatek nas ne zanima
           --teza,                           ta podatek nas ne zanima
         davek,
         posiljateljev_id)

 SELECT inserted.sifra,
           inserted.naziv,
           inserted.cena,
           inserted.kolicina,
 inserted.barva,
           --inserted.skladisce_id,           ta podatek nas ne zanima
           --inserted.teza,                   ta podatek nas ne zanima
           inserted.davek,
           inserted.id
 FROM inserted
 INNER JOIN [Artikli_vmesna_tabela] AVT ON inserted.id = AVT.posiljateljev_id
 WHERE AVT.id IS NULL
```

2.2.2.2 Prejemnikova procedura

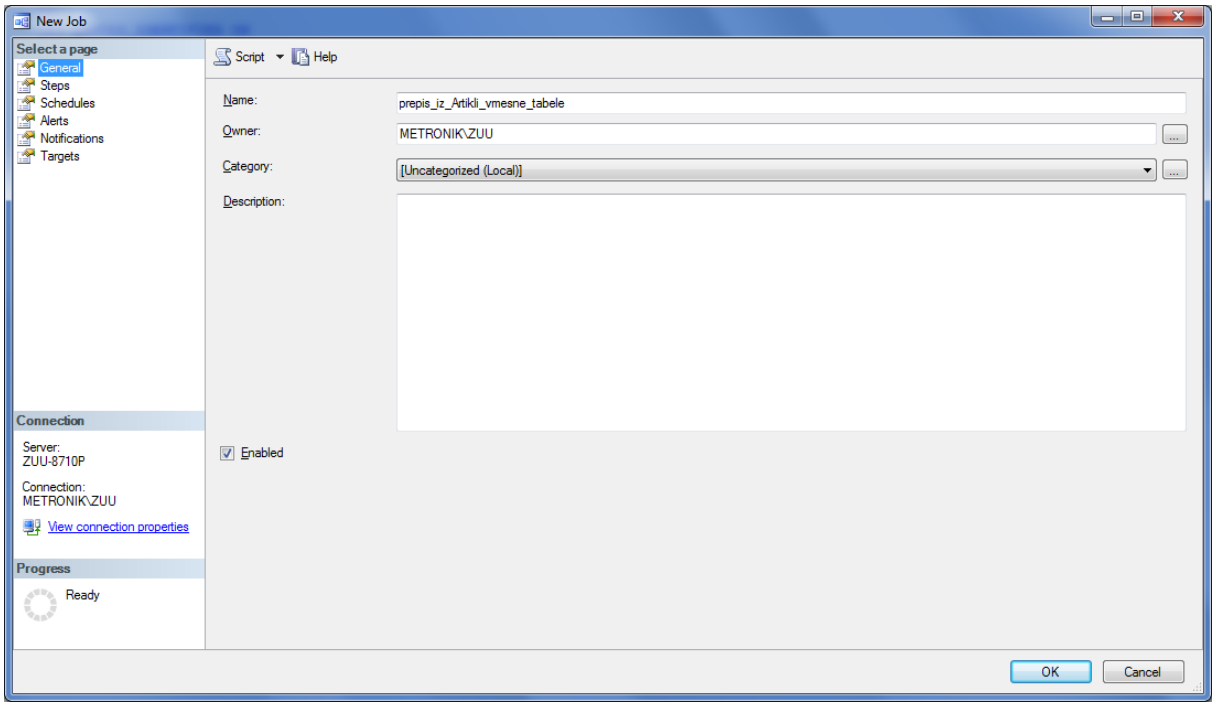
```
CREATE PROCEDURE [dbo].[Artikli_na_prejemniku_upd_ins]

 AS

INSERT INTO [mepis_ls].[dbo].[Artikli_na_prejemniku]
        ([sifra]
        ,[naziv]
        ,[cena]
        ,[barva]
        --,[skladisce_id] ne rabimo, dodamo id svojega skladišča
        ,[teza]
        ,[zunanji_id])
    (SELECT AVT.[sifra]
                    ,AVT.[naziv]
                    ,AVT.[cena]
                    --,[kolicina]
                    ,AVT.[barva]
                    ,AVT.[teza]
                    --,[skladiscna_lokacija]
                    ,AVT.[posiljateljev_id]
  FROM [dbo].[Artikli_vmesna_tabela_view] AS AVT
  LEFT JOIN [dbo].[Artikli_na_prejemniku] ANP ON AVT.posiljateljev_id =
ANP.zunanji_id
  WHERE AVT.id IS NULL)
GO
```



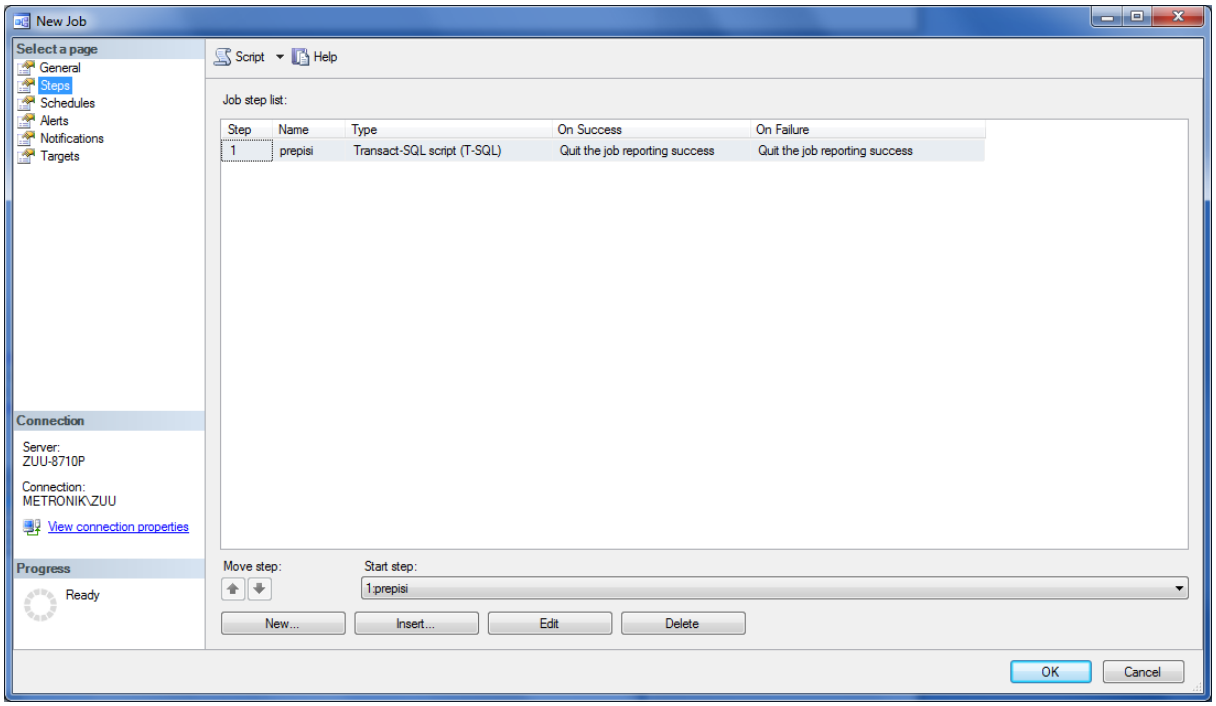
Slika 3: Konceptualni model - vmesna tabela

2.2.2.3 Nastavitve opravila

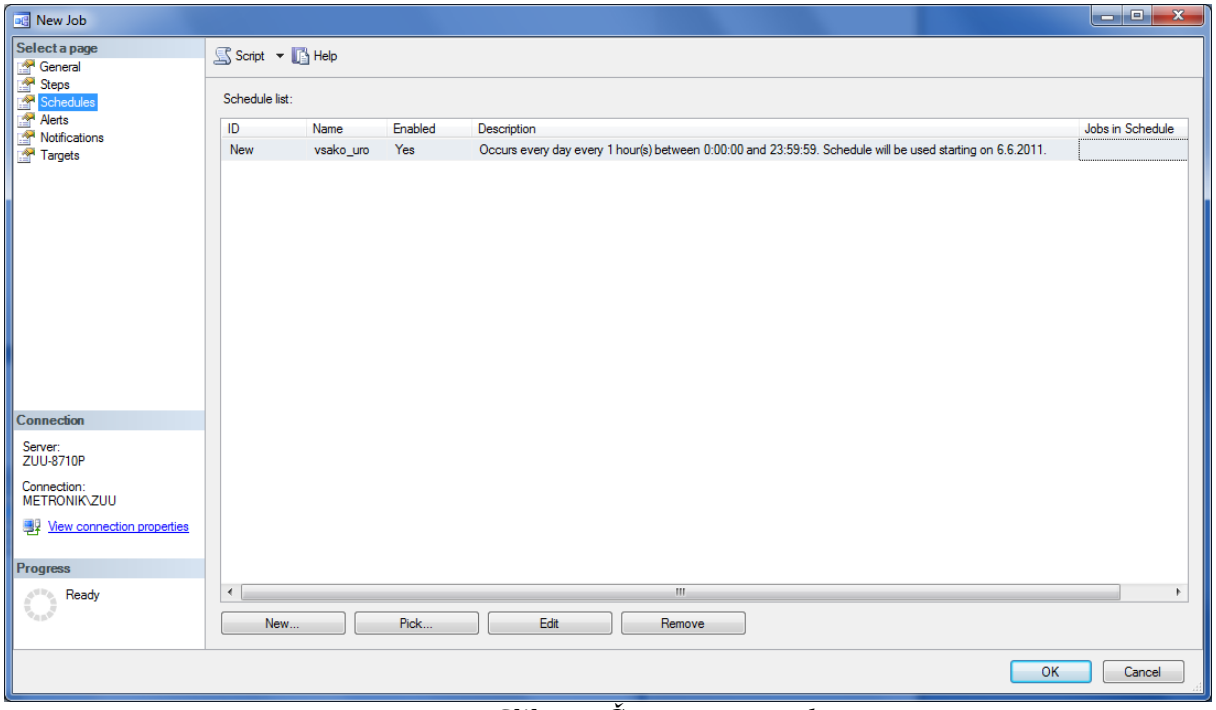
Opravilo (Slika 4) se uporablja zato, da se procedura, ki sem jo napisal zgoraj, zaganja avtomatsko. Na opravilu (Slika 5) nastavimo korake, katero proceduro bomo zaganjali. V naslednjem koraku je potrebno nastaviti urnik (Slika 6), kdaj se bo zaganjala. Jaz sem ga nastavil tako, da se zaganja vsak dan, vsako uro. Če je tabela bolj prometna, je potrebno nastaviti bolj pogosto izvajanje procedure, odvisno od tega, kako ažurne podatke potrebujemo.



Slika 4: Kreacija opravila



Slika 5: Vpisovanje korakov



Slika 6: Časovni interval zagona

2.2.3 Kreacija SB procedure

2.2.3.1 Aktivacija SB-ja

```
USE master;
GO
ALTER DATABASE mepis_ls
    SET ENABLE_BROKER;
GO
USE mepis_ls;
GO
```

2.2.3.2 Kreacija za tipe sporočil

```
CREATE MESSAGE TYPE
    [//AWDB/1DBSample/ZahtevajSporocilo]
    VALIDATION = WELL_FORMED_XML;
CREATE MESSAGE TYPE
    [//AWDB/1DBSample/PonovnoPosljiSporocilo]
    VALIDATION = WELL_FORMED_XML;
GO
```

2.2.3.3 Kreacija pogodbe

```
CREATE CONTRACT [//AWDB/1DBSample/Posiljanje63040183]
    ([//AWDB/1DBSample/ZahtevajSporocilo]
    SENT BY INITIATOR,
    [//AWDB/1DBSample/PonovnoPosljiSporocilo]
    SENT BY TARGET
    );
```

2.2.3.4 Kreiranje Prejemnikove vrste in servica

```
CREATE QUEUE PrejemnikovaVrsta;

CREATE SERVICE
    [//AWDB/1DBSample/PrejemnikovService]
    ON QUEUE PrejemnikovaVrsta
    ([//AWDB/1DBSample/Posiljanje63040183]);
```

2.2.3.5 Pošiljateljeva vrsta

```
CREATE QUEUE posiljateljevaVrsta;

CREATE SERVICE
    [//AWDB/1DBSample/PosiljateljevService]
    ON QUEUE posiljateljevaVrsta;
```

2.2.3.6 Pošiljanje sporočila

```
DECLARE @dialog UNIQUEIDENTIFIER;
DECLARE @ZahtevanoSporocilo NVARCHAR(100);

BEGIN TRANSACTION;

BEGIN DIALOG @dialog
    FROM SERVICE
     [//AWDB/1DBSample/PosiljateljevService]
    TO SERVICE
     N'//AWDB/1DBSample/PrejemnikovService'
    ON CONTRACT
     [//AWDB/1DBSample/Posiljanje63040183]
    WITH
       ENCRYPTION = OFF;

SELECT @ZahtevanoSporocilo =
      N'<RequestMsg>Message for Target service.</RequestMsg>';

SEND ON CONVERSATION @dialog
    MESSAGE TYPE
    [//AWDB/1DBSample/ZahtevajSporocilo]
    (@ZahtevanoSporocilo);

SELECT @ZahtevanoSporocilo AS SentRequestMsg;

COMMIT TRANSACTION;
GO
```

2.2.3.7 Ponovno pošiljanje sporočila

```
DECLARE @ponovnoPosiljanje UNIQUEIDENTIFIER;
DECLARE @Sporocilo NVARCHAR(100);
DECLARE @NazivSporocila sysname;

BEGIN TRANSACTION;

WAITFOR
( RECEIVE TOP(1)
   @ponovnoPosiljanje = conversation_handle,
   @Sporocilo = message_body,
   @NazivSporocila = message_type_name
  FROM PrejemnikovaVrsta
), TIMEOUT 1000;

SELECT @Sporocilo AS ReceivedRequestMsg;
print @NazivSporocila
IF @NazivSporocila =
  N'//AWDB/1DBSample/ZahtevajSporocilo'
```

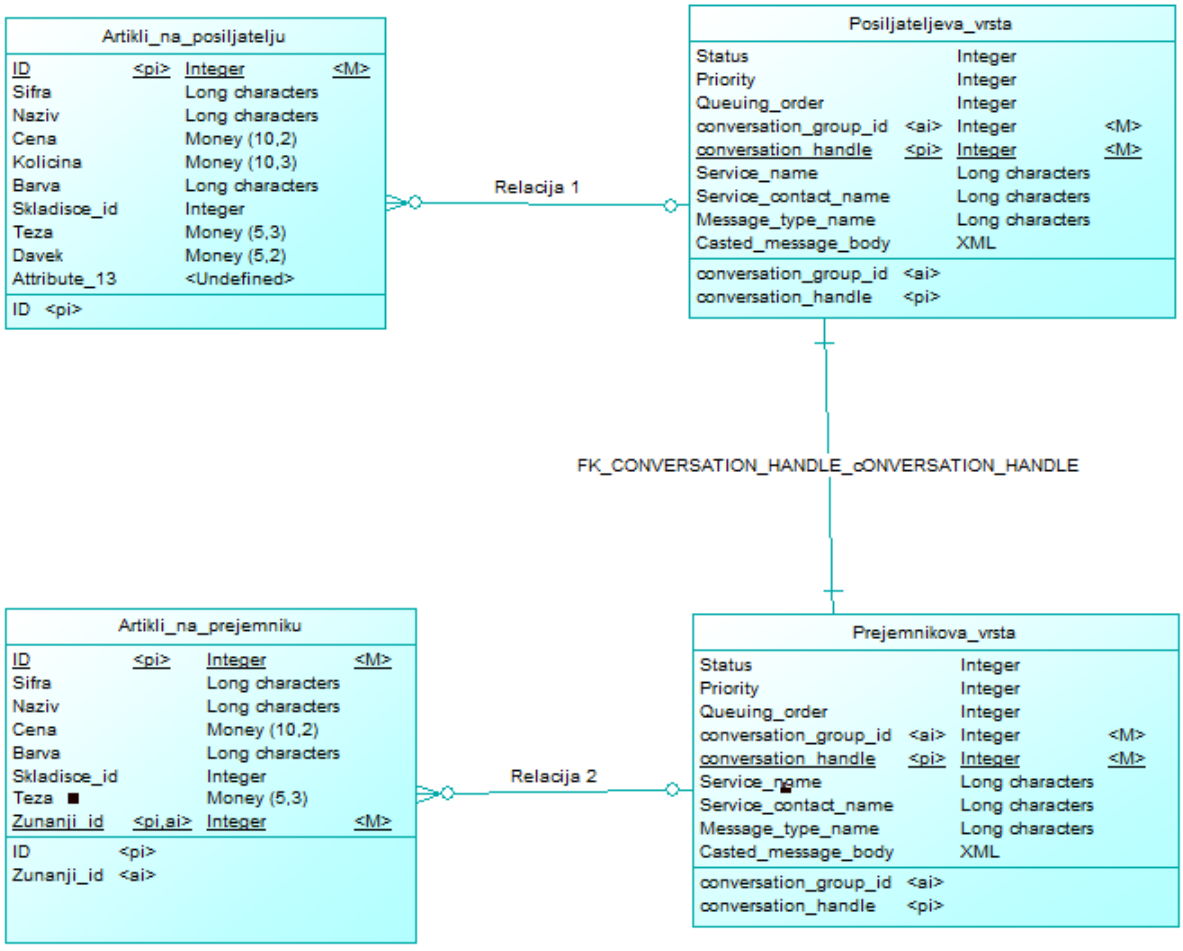
```
BEGIN
    DECLARE @ZahtevanoSporocilo NVARCHAR(100);
    SELECT @ZahtevanoSporocilo =
    N'<ReplyMsg>Message for Initiator service.</ReplyMsg>';

    SEND ON CONVERSATION @ponovnoPosiljanje
        MESSAGE TYPE
        [//AWDB/1DBSample/PonovnoPosljiSporocilo]
        (@ZahtevanoSporocilo);

    END CONVERSATION @ponovnoPosiljanje;
END

SELECT @ZahtevanoSporocilo AS SentReplyMsg;

COMMIT TRANSACTION;
GO
```

2.2.3.8 Prejetje ponovno poslanega sporočila

```
DECLARE @sporocilo NVARCHAR(100);
DECLARE @dialog UNIQUEIDENTIFIER;

BEGIN TRANSACTION;

WAITFOR
( RECEIVE TOP(1)
   @dialog = conversation_handle,
   @sporocilo = message_body
  FROM posiljateljevaVrsta
), TIMEOUT 1000;

END CONVERSATION @dialog;

SELECT @sporocilo AS ReceivedReplyMsg;

COMMIT TRANSACTION;
```

2.2.3.9 Opis pošiljanja sporočila s SB-jem

V zgornjih primerih sem prikazal, kako lahko na dva načina prepišemo podatke iz ene baze v drugo. Pri obeh načinih je rezultat enak, le pošiljanje podatkov se razlikuje. Za pošiljanje podatkov v vrsti uporabljamo sprožilec in opravilo. Če opravilo ni dobro nastavljeno, se nam lahko velikokrat ustavi in se nam podatki ne osvežujejo več. Takemu primeru se je nemogoče izogniti. Omilimo ga lahko tako, da nastavimo opozorilo, da je opravilo prenehalo z delovanjem (opravilo nam ga pošlje). Tudi SB ima lahko podobne težave, vendar se sporočila shranijo v vrsti in čakajo, dokler jih ne obdelamo ali izbrišemo. S SB-jem se izognemo sprožilcu in opravilu. S proceduro zaženemo vpis v vrsto, ki prepiše podatke glede na pogodbo (izognemo se sprožilcu) iz pošiljateljeve vrste v prejemnikovo vrsto. V prejemnikovi vrsti se ob vsakem vpisu sproži procedura (izognemo se opravilu), ki izpiše podatke iz XML dokumenta v ustrezne tabele. Pri izpustitvi dveh nepotrebnih korakov se nam vzdrževanje prenosov podatkov zmanjša.



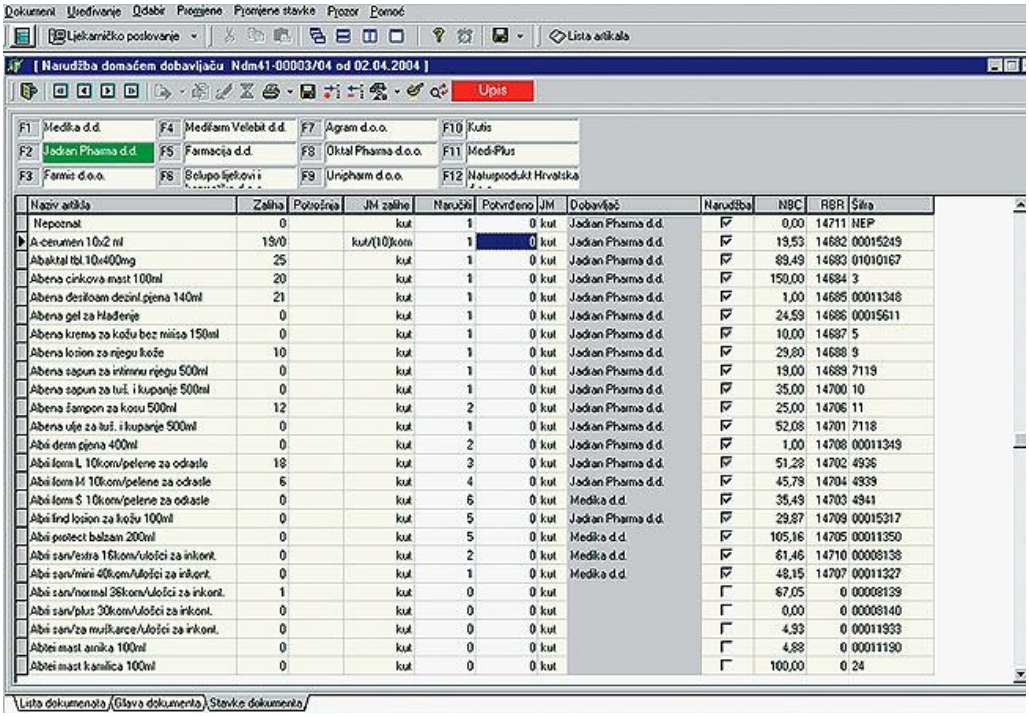
Slika 7: Konceptualni model Service Broker-ja

2.3 Uporaba SB–ja na primeru v praksi

2.3.1 Predstavitev razlogov za uporabo SB-ja v praksi

JGL je farmacevtsko podjetje, ki se ukvarja z generičnimi farmacevtskimi proizvodi. V podjetju se že dlje časa uporablja ERP informacijski sistem IPS (Slika 8), ki se uporablja za izdajo delovnih nalogov ter dokumentov za izdajo surovin iz skladišča. Ta informacijski sistem deluje zelo površno, saj samo natisne dokument, kaj je potrebno izdati, ne vodi pa, kaj se s tem materialom dogaja. S tem informacijskim sistemom upravljajo samo vodje oddelkov, ne pa tudi zaposleni v skladišču oz. proizvodnji. Zaposleni uporabljajo natisnjene dokumente.

Če povzamemo delovanje informacijskega sistema, deluje v naslednjem vrstnem redu. Vodja v IPS vpiše potrebne podatke, jih izpiše in jih preda delavcu v skladišču. Ta delavec glede na dokument izda potreben material, ga pripelje na določeno mesto in dokument ustrezno izpolnjen vrne vodji. Ta to vpiše nazaj v program IPS. Delo je zelo zamudno, saj vsebuje prepisovanje podatkov nazaj v računalnik, kjer pa lahko velikokrat pride do človeških napak pri prepisovanju.

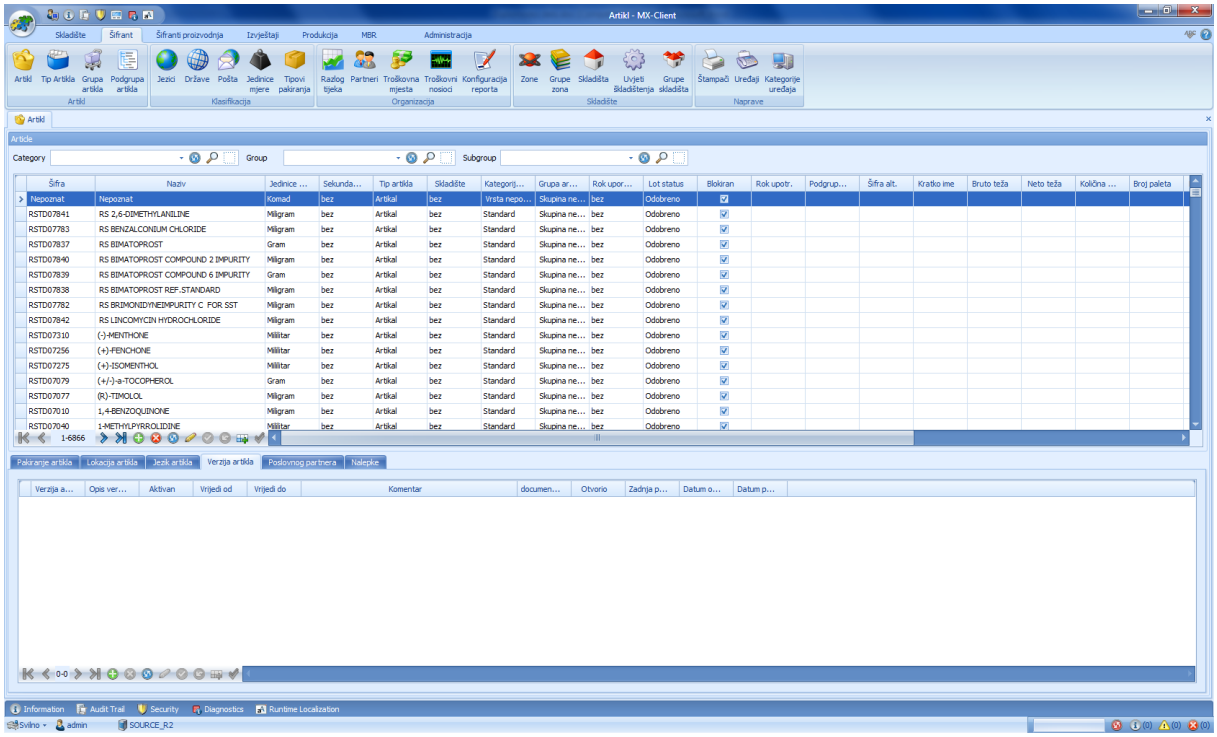


Slika 8: IPS informacijski sistem [2]

Zato so se v JGL-ju odločili, da vzpostavijo informacijski sistem, ki beleži, kaj se z izdanim materialom dogaja tekom proizvodnje ter avtomatizira proces vnosa dokumentov.

Tako so se odločili za naš informacijski sistem MePIS (Slika 9). To je informacijski sistem, ki skrbi za obvladovanje proizvodnje z ravni delovnih nalogov in obvladovanje materialnih tokov v proizvodnji, vključno s proizvodnimi skladišči. Poleg tega MePIS nudi tudi podporo za obvladovanje učinkovitosti procesov in kakovosti proizvodov. MePIS odlikuje tesna integracija s procesnimi sistemi, tako da modeli poslovnega odločanja v MePIS-u temeljijo na dejanskih podatkih iz procesa v najvišji možni meri. Po drugi strani se MePIS učinkovito

povezuje s sistemi ERP in na ta način prispeva k celovitemu pretoku informacij v proizvodnem podjetju.



Slika 9: MePIS informacijski sistem [3]

Ker sta MePIS in IPS dva ločena, tuja sistema, in ker si izmenjujeta podatke, je bilo potrebno narediti povezavo med njima. Obema sistemoma je skupen PB SQL Server, zato smo se odločili za integracijo sistemov z uporabo SB-ja. Za alternativo SB-ju smo imeli vmesno bazo, v katero bi se kopirali podatki tako iz sistema IPS kot tudi iz sistema MePIS. Tukaj pride še do enega problema, saj je potrebno na novo postaviti SQL Server, kjer pride do vmesnega člena med dvema bazama. Tukaj ni direktne komunikacije. Poleg tega smo vmesno bazo ovrgli še zaradi dveh drugih pomanjkljivosti in sicer:

- Varnostne pomanjkljivosti. Podatki niso zavarovani oz. zakodirani, ni garancije, da ni tretja oseba prebrala teh zapisov.
- Prepis podatkov (potrebna opravila). Potrebno je napisati vsako opravilo za vsako tabelo, ki se prepisuje; ni zagotovila, da se bodo vsi podatki prenesli. Opravila se lahko ustavijo. Prepisujejo se cele tabele in ne samo tisti zapisi, ki so novi.

2.3.2 Zakaj pravzaprav sploh potrebujemo SB?

Za prenos podatkov med dvema informacijskima sistemoma smo uporabili SB. Ta skrbi, da so podatki v obeh sistemih ažurni in konsistentni. S SB-jem imamo garantiran prenos podatkov, saj se podatki shranjujejo v vrstah in se pošiljajo vse dokler niso poslani.

IPS je informacijski sistem, ki deluje na način » Klient - Strežnik«, podatki so zapisani v centralni bazi tipa MS SQL Server, ki se bodo izmenjavali z MePIS-om na nivoju MS SQL Server-ja.
V IPS-u se določijo vsi šifranti, ki se bodo uporabljali v MePIS-u in se preko SB-ja prepišejo v MePIS-ovo bazo. Kakršna koli sprememba se naredi na MePIS-ovih šifrantih, se nemudoma prenese preko SB-ja nazaj v IPS-ovo bazo. Tako so podatki ves čas ažurni ter konsistentni.

2.3.3 Struktura sistema ali okolja?

Podjetje JGL uporablja poslovni sistem IPS. Z informacijskim sistemom MePIS je povezan na dveh nivojih in sicer:

- na nivoju šifrantov
- ter na nivoju dokumentov.

Preden se je začel sestavljati vmesnik med IPS-om in MePIS-om, se je bilo potrebno dogovoriti, katera polja se bodo prenašala. Tako je nastala funkcijska dokumentacija, v kateri so napisana polja, ki se prepišejo iz IPS-a v MePIS.

V tabeli 1 prikazujem podatke o Enoti mere. Leva stran opisuje polje v tabeli »custom_jinfo.Jed_mjer« v IPS-u , desna stran opisuje polje v tabeli »types.measure_unit« v MePIS-u.

Tabela 1 Povezava polj med IPS-om in MePIS-om

IPS polje	Vsebina	MePIS LS polje
S_JEDINICE	Šifra „Id“ enota mere v sistemu IPS. Vrednost podatka je celo število. Vrednost je unikatna.	external_id
N_JEDINICE	Naziv enote	name
KRATICA	Kratica enote mere (kg, l, m, …)	code
VELIČINA		short_name
AKTIVAN	Tekst (2 znaka) „Da“/„Ne“ - „Da“: aktivna enota mere. - „Ne“: neaktivna enota mere.	record_status - 0: aktivna - 1: neaktivna
LAST_PROM	Datum in čas zadnje spremembe.	record_timestamp (insert), record_timestamp_upd (update)
S_OSOBE	ID osebe iz IPS-a, ki je spremenila zapis.	External_user_id

Trenutno se v skladiščnem delu med sistemoma izmenjuje 23 tabel šifrantov in prepisuje več kot 24 dokumentov. Dnevno se v IPS-u kreira približno 50 dokumentov. Zapre se jih približno 40. Vsaka sprememba oz. zapis se zaradi ažurnosti mora prepisati v drug sistem. Ko se sproži zapis v tabelo, ga SB pošlje v vrsto, od tu naprej se pošlje v vrsto prejemnika, kjer se podatki procesirajo in vpišejo v tabelo.

Struktura dokumenta je drugačna. Dokument je sestavljen iz treh delov. Prvi del je glava dokumenta. Vsaka glava se razlikuje glede na namen dokumenta. Predstavil bom dokument »Primka« .

V glavi so podatki o:

- tipu dokumenta,
- lokaciji in skladišču sestavin na dokumentu,
- datumu kreacije,
- številki dokumenta,
- družabniku, ki je priskrbel sestavine
- in ostali podatki, ki so pomembni za povezavo med dokumenti.

Drugi del dokumenta so »stavke« oziroma deli dokumenta, na katerem so zabeleženi artikli, ki se nahajajo na tem dokumentu: artiklov je lahko poljubno mnogo.

Na postavi so naslednja polja:

- številka postavke,
- lokacija in skladišče,
- artikel,
- verzija artikla,
- količina artikla,
- količina pakiranja artikla,
- kontrolna številka,
- serijska številka,
- rok uporabe,
- in ostala polja, ki se uporabljajo za pravilnost uporabe dokumenta.

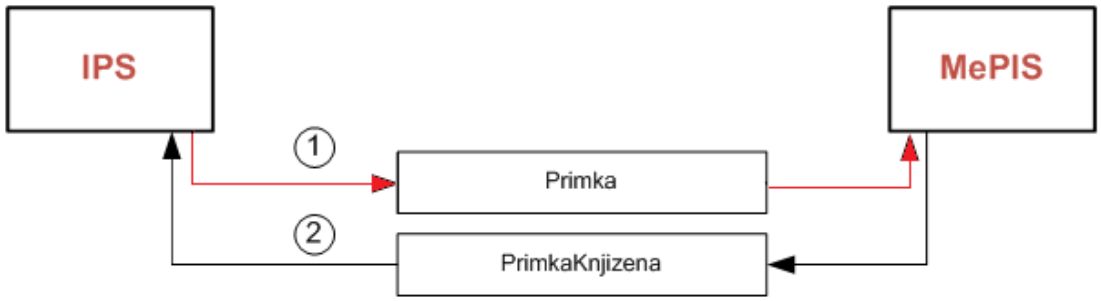
V tretjem delu so pakiranja. Pakiranja se ne prenašajo iz IPS-a, temveč se vpišejo v MePIS-u. Vsak artikel je lahko sestavljen iz poljubno mnogo pakiranj. Količina postavk dokumenta se mora ujemati s seštevkom količin pakiranja. Npr., če imamo napisano v postavki količino 80, in pakiranje artikla po 10 komadov, mora biti v pakiranju 8 x po 10 kosov določenega artikla. V pakiranju se še doda logistična enota (paleta, karton, sod …).

Primer XML dokumenta „Primka“:

```
<Primka>
 <Int_BrDok>2</Int_BrDok>
 <S_Lokacije>1</S_Lokacije>
 <S_skladista>1</S_skladista>
 <Godina>2010</Godina>
 <Broj_dok>Primka339-672/10</Broj_dok>
 <S_GrupeDok>1</S_GrupeDok>
 <Tip_Dok>1</Tip_Dok>
 <S_Partnera>2438</S_Partnera>
 <Datum_dok>2010-10-31T00:00:00</Datum_dok>
 <Vezni_dok>INV-91270232</Vezni_dok>
 <Dat_VezDok>2009-12-22T00:00:00</Dat_VezDok>
 <Last_prom>2010-01-04T08:02:00</Last_prom>
 <S_operatera>10</S_operatera>
 <Stavke>
  <Stavka>
   <Br_stavke>7</Br_stavke>
   <S_Lokacije>1</S_Lokacije>
   <Godina>2010</Godina>
   <S_skladista>1</S_skladista>
   <Int_BrDok>2</Int_BrDok>
   <S_artikla>725</S_artikla>
   <Kolicina>100.000000</Kolicina>
   <Broj_Pakovanja>10</Broj_Pakovanja>
   <Kontrolni_Broj>SS_1</Kontrolni_Broj>
   <Serijski_Broj>LR</Serijski_Broj>
```

```
    <VrijediDo>2013-02-02T00:00:00</VrijediDo>
    <Broj_Verzije />
    <Last_prom>2010-01-04T08:02:00</Last_prom>
    <S_operatera>10</S_operatera>
   </Stavka>
  </Stavke>
</Primka>
```

Ko se izpolnijo vsi podatki, se lahko dokument zaključi. Ob pritisku gumba »knjiži« pokliče procedura, ki preveri, če so vsi potrebni podatki vpisani in pravilni. Po potrditvi se spremeni status dokumenta ter se pošlje s SB-jem v IPS (Slika 10).



Slika 10 Prikaz toka informacij iz IPS-a v MePIS

2.3.4 Primer XML Dokumenta dobljenega sporočila iz IPS-a in obdelava le-tega

Primer prejema sporočil iz IPS-a (Slika 10); pot prve puščice.
Primer dobljenega sporočila »Primke« iz IPS-a:

```
<Primka xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Int_BrDok>3</Int_BrDok>
  <S_Lokacije>1</S_Lokacije>
  <S_skladista>1</S_skladista>
  <Godina>2011</Godina>
  <Broj_dok>Pri11-00003/11</Broj_dok>
  <S_GrupeDok>1</S_GrupeDok>
  <Tip_Dok>1</Tip_Dok>
  <S_Partnera>164</S_Partnera>
  <Datum_dok>2011-01-03T00:00:00</Datum_dok>
  <Vezni_dok>R-374/2010</Vezni_dok>
  <Dat_VezDok>2010-12-29T00:00:00</Dat_VezDok>
  <Last_prom>2011-01-03T08:22:00</Last_prom>
  <S_operatera>22</S_operatera>
  <Stavke xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Stavka>
      <Br_stavke>3</Br_stavke>
      <S_Lokacije>1</S_Lokacije>
      <Godina>2011</Godina>
      <S_skladista>1</S_skladista>
      <Int_BrDok>3</Int_BrDok>
      <S_artikla>3188</S_artikla>
      <Kolicina>22700.000000</Kolicina>
      <Broj_Pakovanja>2</Broj_Pakovanja>
      <Kontrolni_Broj>A - 0002/11</Kontrolni_Broj>
      <Serijski_Broj>1032/10</Serijski_Broj>
      <VrijediDo>2013-01-03T00:00:00</VrijediDo>
      <Broj_Verzije xsi:nil="true" />
      <S_proizvodjaca>164</S_proizvodjaca>
      <Last_prom>2011-01-03T08:22:00</Last_prom>
      <S_operatera>22</S_operatera>
    </Stavka>
    <Stavka>
      <Br_stavke>4</Br_stavke>
      <S_Lokacije>1</S_Lokacije>
      <Godina>2011</Godina>
      <S_skladista>1</S_skladista>
      <Int_BrDok>3</Int_BrDok>
      <S_artikla>3208</S_artikla>
      <Kolicina>5000.000000</Kolicina>
      <Broj_Pakovanja>1</Broj_Pakovanja>
      <Kontrolni_Broj>A - 0003/11</Kontrolni_Broj>
      <Serijski_Broj>1029/10</Serijski_Broj>
      <VrijediDo>2013-01-03T00:00:00</VrijediDo>
      <Broj_Verzije xsi:nil="true" />
```

```
    <S_proizvodjaca>164</S_proizvodjaca>
    <Last_prom>2011-01-03T08:22:00</Last_prom>
    <S_operatera>22</S_operatera>
   </Stavka>
   <Stavka>
    <Br_stavke>5</Br_stavke>
    <S_Lokacije>1</S_Lokacije>
    <Godina>2011</Godina>
    <S_skladista>1</S_skladista>
    <Int_BrDok>3</Int_BrDok>
    <S_artikla>5994</S_artikla>
    <Kolicina>22500.000000</Kolicina>
    <Broj_Pakovanja>15</Broj_Pakovanja>
    <Kontrolni_Broj>A - 0004/11</Kontrolni_Broj>
    <Serijski_Broj>1121/10</Serijski_Broj>
    <VrijediDo>2013-11-03T00:00:00</VrijediDo>
    <Broj_Verzije xsi:nil="true" />
    <S_proizvodjaca>164</S_proizvodjaca>
    <Last_prom>2011-01-03T08:22:00</Last_prom>
    <S_operatera>22</S_operatera>
   </Stavka>
   <Stavka>
    <Br_stavke>6</Br_stavke>
    <S_Lokacije>1</S_Lokacije>
    <Godina>2011</Godina>
    <S_skladista>1</S_skladista>
    <Int_BrDok>3</Int_BrDok>
    <S_artikla>5995</S_artikla>
    <Kolicina>50000.000000</Kolicina>
    <Broj_Pakovanja>5</Broj_Pakovanja>
    <Kontrolni_Broj>A - 0005/11</Kontrolni_Broj>
    <Serijski_Broj>1123/10</Serijski_Broj>
    <VrijediDo>2013-11-03T00:00:00</VrijediDo>
    <Broj_Verzije xsi:nil="true" />
    <S_proizvodjaca>164</S_proizvodjaca>
    <Last_prom>2011-01-03T08:22:00</Last_prom>
    <S_operatera>22</S_operatera>
   </Stavka>
  </Stavke>
</Primka>
```

Zgornje XML sporočilo, dobljeno iz vrste, ki ga je poslal IPS, se preko procedur v SQL Server-ju obdela in vpiše v tabeli warehouse.document_header in [warehouse].[document_detail]. Ko so podatki v tabeli, čakajo na ureditev. Te jih preko aplikacije uredijo, dopišejo in spremenijo ter jih ob zaključku dokumenta pošljejo nazaj v IPS (Slika 11).

2.3.5 Primer dobljenega poročila poslanega iz MePis-a v IPS

Slika 11 Prikaz toka informacij iz MePIS-a v IPS

```
<Primka xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Int_BrDok>3</Int_BrDok>
  <S_Lokacije>1</S_Lokacije>
  <S_skladista>1</S_skladista>
  <Godina>2011</Godina>
  <Broj_dok>Pri11-00003/11</Broj_dok>
  <S_GrupeDok>1</S_GrupeDok>
  <Tip_Dok>1</Tip_Dok>
  <S_Partnera>164</S_Partnera>
  <Datum_dok>2011-01-03T00:00:00</Datum_dok>
  <Vezni_dok>R-374/2010</Vezni_dok>
  <Dat_VezDok>2010-12-29T00:00:00</Dat_VezDok>
  <Last_prom>2011-01-03T08:22:00</Last_prom>
  <S_operatera>22</S_operatera>
  <Stavke xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Stavka>
      <Br_stavke>3</Br_stavke>
      <S_Lokacije>1</S_Lokacije>
      <Godina>2011</Godina>
      <S_skladista>1</S_skladista>
      <Int_BrDok>3</Int_BrDok>
      <S_artikla>3188</S_artikla>
      <Kolicina>22700.000000</Kolicina>
      <Broj_Pakovanja>2</Broj_Pakovanja>
      <Kontrolni_Broj>A - 0002/11</Kontrolni_Broj>
      <Serijski_Broj>1032/10</Serijski_Broj>
      <VrijediDo>2013-01-03T00:00:00</VrijediDo>
      <Broj_Verzije xsi:nil="true" />
      <S_proizvodjaca>164</S_proizvodjaca>
      <Last_prom>2011-01-03T08:22:00</Last_prom>
      <S_operatera>22</S_operatera>
    </Stavka>
    <Stavka>
      <Br_stavke>4</Br_stavke>
      <S_Lokacije>1</S_Lokacije>
```

```
    <Godina>2011</Godina>
    <S_skladista>1</S_skladista>
    <Int_BrDok>3</Int_BrDok>
    <S_artikla>3208</S_artikla>
    <Kolicina>5000.000000</Kolicina>
    <Broj_Pakovanja>1</Broj_Pakovanja>
    <Kontrolni_Broj>A - 0003/11</Kontrolni_Broj>
    <Serijski_Broj>1029/10</Serijski_Broj>
    <VrijediDo>2013-01-03T00:00:00</VrijediDo>
    <Broj_Verzije xsi:nil="true" />
    <S_proizvodjaca>164</S_proizvodjaca>
    <Last_prom>2011-01-03T08:22:00</Last_prom>
    <S_operatera>22</S_operatera>
   </Stavka>
   <Stavka>
    <Br_stavke>5</Br_stavke>
    <S_Lokacije>1</S_Lokacije>
    <Godina>2011</Godina>
    <S_skladista>1</S_skladista>
    <Int_BrDok>3</Int_BrDok>
    <S_artikla>5994</S_artikla>
    <Kolicina>22500.000000</Kolicina>
    <Broj_Pakovanja>15</Broj_Pakovanja>
    <Kontrolni_Broj>A - 0004/11</Kontrolni_Broj>
    <Serijski_Broj>1121/10</Serijski_Broj>
    <VrijediDo>2013-11-03T00:00:00</VrijediDo>
    <Broj_Verzije xsi:nil="true" />
    <S_proizvodjaca>164</S_proizvodjaca>
    <Last_prom>2011-01-03T08:22:00</Last_prom>
    <S_operatera>22</S_operatera>
   </Stavka>
   <Stavka>
    <Br_stavke>6</Br_stavke>
    <S_Lokacije>1</S_Lokacije>
    <Godina>2011</Godina>
    <S_skladista>1</S_skladista>
    <Int_BrDok>3</Int_BrDok>
    <S_artikla>5995</S_artikla>
    <Kolicina>50000.000000</Kolicina>
    <Broj_Pakovanja>5</Broj_Pakovanja>
    <Kontrolni_Broj>A - 0005/11</Kontrolni_Broj>
    <Serijski_Broj>1123/10</Serijski_Broj>
    <VrijediDo>2013-11-03T00:00:00</VrijediDo>
    <Broj_Verzije xsi:nil="true" />
    <S_proizvodjaca>164</S_proizvodjaca>
    <Last_prom>2011-01-03T08:22:00</Last_prom>
    <S_operatera>22</S_operatera>
   </Stavka>
  </Stavke>
</Primka>
```

2.3.6 Potek sporočila iz enega v drug sistem preko SB-ja

V nadaljevanju bomo pogledali, kako poteka celotna pot prenosa polj dokumenta »primka« iz enega sistema v drugega. Začeli bomo z vnosom v sistem IPS. Ko uporabnik razpiše delovni nalog »primka«, se na koncu procedure zažene klic procedure »ustvari_primku_ins«. Ta procedura iz zapisa v tabeli ustvari sporočilo v XML obliki, ki ga pošlje v vrsto. Preden sporočilo pošlje v vrsto, se doda dialog, ki vsebuje: iz katerega in v kateri servis se pošilja. Pri tem še doda pogodbo, ki je nujna za identifikacijo pogovora. Po končani sestavi sporočila se vse skupaj zapiše v vrsto, kot zapis v tabelo.

V vrsti se sproži povezava na SB v podatkovni bazi sistema MePIS. Če je Servis vklopljen, in če je pogodba enaka, kot smo se predhodno dogovorili, se podatki prepišejo v prejemnikovo vrsto na sistemu MePIS. Ko je sporočilo enkrat v vrsti, se sproži procedura [custom_jgl].[sb_ProcessRequestMessages]. Ta procedura prebere prvo vrstico iz vrste in preveri, če so podatki pravilni.

```
RECEIVE TOP(1)
@ch = [conversation_handle],
@messagetypename = message_type_name,
        @messagebody = CAST(message_body AS XML),
        @servicename = [service_name],
        @contractname = service_contract_name
        FROM dbo.Metronik_queue

IF (@messagetypename = 'Primka')or
(@messagetypename='AnalizaSirovina')or
(@messagetypename='RashodAnalitika') or
(@messagetypename = 'PreskladistenjeAnalitika')or
(@messagetypename = 'OdobrenjeSirovina')or
(@messagetypename='Rashod')or
(@messagetypename='Preskladistenje')or
(@messagetypename='Izdatnica')or
(@messagetypename='IzdatnicaRadniNalog')or (@messagetypename='OdobrenjeProizvoda')
BEGIN
```

Pri preverjanju pravilnosti podatkov preveri, če smo poslali pravilen tip sporočil (messagetypename). Nato prepiše najnujnejše podatke v evidenčno tabelo s prometom, da bomo v prihodnosti vedeli, kateri podatki so se prenesli iz pošiljateljeve vrste. Na koncu bomo še dodali status ali je bilo sporočilo uspešno dodano v prejemnikovo tabelo. Po opravljenem vpisu v evidenčno tabelo se zažene procedura sb_Process_Received_Messages_Primka. Procedura z razčlenjevalnikom kode (Parser) razčleni XML dokument. Razčlenitev opravi postopoma. Začne s prvim vozliščem (Node), katere spremenljivke shrani v parametre, ki jih kasneje preveri in vpiše v »primko«.

```
select @code = case when len(RTRIM(dok.col.value('./Broj_dok[1]','VARCHAR(30)')))=0
then null else dok.col.value('./Broj_dok[1]','VARCHAR(30)') end,
 @s_partnera = case when len(rtrim(dok.col.value('./S_Partnera[1]','varchar(30)'))) = 0 then
null else dok.col.value('./S_Partnera[1]','int') end,
 @start_object_id = bp.id,
          @s_skladista = case when
```

```
LEN(RTRIM(dok.col.value('./S_skladista[1]','varchar(30)')))=0 then null else
dok.col.value('./S_skladista[1]','int') end,
          @end_object_id = w.id,
from @message.nodes('/Primka') DOK(COL)
        left join types.business_partner bp on dok.col.value('./S_Partnera[1]','int') =
bp.external_id
        left join types.warehouse w on dok.col.value('./S_skladista[1]','int') = w.external_id
```

Naslednji korak preverjanja primke je zapisovanje listov (leafs) v začasno tabelo. Po vpisu se preveri pravilnost le-teh. Na koncu se zapisi vpišejo v tabelo document_detail, ter vpiše id vozlišča, da vemo kateremu vozlišču zapis pripada.

```
insert into @lines
select case when len(RTRIM(dok.col.value('./Br_stavke[1]','varchar(20)')))=0 then null else
dok.col.value('./Br_stavke[1]','int') end AS external_id,
                  w.id as end_object_id,
                  ta.id as article_id,
                  (case when rtrim(dok.col.value('./Broj_Pakovanja[1]','nvarchar(128)')) = ''
then 0
else dok.col.value('./Broj_Pakovanja[1]','decimal(18,6)') end) as packaging_quantity,
                  case when
len(RTRIM(dok.col.value('./Kontrolni_Broj[1]','nVARCHAR(128)')))=0 then null else
rtrim(dok.col.value('./Kontrolni_Broj[1]','nVARCHAR(128)')) end as material_cn,
                   case when
len(RTRIM(dok.col.value('./Serijski_Broj[1]','nVARCHAR(32)')))=0 then null else
rtrim(dok.col.value('./Serijski_Broj[1]','nVARCHAR(32)')) end as serial_cn,
                   (case when rtrim(dok.col.value('./VrijediDo[1]','nvarchar(30)'))='' then null
                                 else dok.col.value('./VrijediDo[1]','date') end) as best_before,
                    (case when rtrim(dok.col.value('./Kolicina[1]','nvarchar(30)'))='' then 0
from @message.nodes('/Primka/Stavke/Stavka') DOK(COL)
        left join types.warehouse w on dok.col.value('./S_skladista[1]','int') = w.external_id
        left join types.article ta on dok.col.value('./S_artikla[1]','int') = ta.external_id
```

Če je uvoz podatkov uspešen, procedura sb_ProcessRequestMessages pogovor zapre in pošiljateljevi vrsti pošlje ENDTYPDIALOG. V nasprotnem primeru se pošlje ERROR. V »end dialog-u« spremenimo status v evidenčni tabeli, da smo uspešno uvozili sporočilo.

Primer uspešno poslanega sporočila.

```
IF (@messagetypename ='http://schemas.microsoft.com/SQL/ServiceBroker/EndDialog')
BEGIN
-- End the conversation
  update custom_jgl.sb_SendMessages
    set reply_from_ips = 1,
   reply_from_ips_datetime = GETDATE()
   where conversation_id = @ch
-- Naredim se update v tabeli warehouse.document_header
update warehouse.document_header
   set transfer_date = GETDATE(),
```

```
 transfer_status_id = 1
 where id in (select document_header_id
   from custom_jgl.sb_SendMessages
  where conversation_id = @ch)

END CONVERSATION @ch;
END
```

Primer neuspešno poslanega sporočila.

```
IF (@messageTypeName ='http://schemas.microsoft.com/SQL/ServiceBroker/Error')
begin
-- Poslani XML oznaèim kot neposlan in shranim napako v zapis
 update custom_jgl.sb_SendMessages
   set reply_from_ips = 1,
  reply_from_ips_datetime = GETDATE(),
            processed = 0,
            error_msg_from_ips = @messagebody
  where conversation_id = @ch
```

Sedaj lahko vidimo v prejemnikovi aplikaciji podatke, ki smo jih dobili iz pošiljateljevega informacijskega sistema.

2.3.7 Zaključevanje dokumenta in pošiljanje sporočila pošiljatelju

Ko v MePIS-u vpišemo ustrezne podatke o »primki«, jo pošljemo v IPS. To naredimo tako, da spremenimo status dokumentu. Ta preveri, če so vpisani podatki v dokumentu pravilni in nato zažene proceduro custom_jgl.sb_CreateAndSendMessage2IPS.

```
IF CAST(environment.core_setting_value('Service_Broker_Enabled') AS INT)= 1 AND
ISNULL(@send_message_to_erp, 1) = 1
BEGIN
        BEGIN TRY
EXEC custom_jgl.sb_CreateAndSendMessage2IPS @document_header_id,
@document_status_id,
@document_type_id
        END TRY…
```

Na začetku procedura preveri, če je vrsta izključena. V nadaljevanju sprocesiramo želeno sporočilo, pri katerem pokličemo posebno proceduro, ki nam spremeni zapis iz tabelaričnega zapisa v XML zapis. Spreminjanje zapisa sem opisal v poglavju (2.3.8). Sedaj določimo še ustrezne parametre, s katerimi se bo pogovor med dvema vrstama oziroma servisoma identificiral. Nato se sporočilo pošlje v našo vrsto in čaka, da se karseda hitro pošlje v prejemnikovo vrsto. Ko se sporočilo pošlje iz naše vrste, se izbriše in s tem je pošiljanje na naši strani končano.

```
SELECT @queue_enabled = is_enqueue_enabled
FROM   sys.service_queues
where name = 'Metronik_queue'

--Klicanje procedure, za spremembo zapisa iz tabelaričnega v XML zapis
set @msg = custom_jgl.sb_CreateMessage_PrimkaKnjizena (@document_header_id)

        set @messagename = 'PrimkaKnjizena'

        if @queue_enabled = 1
        begin
                begin try
                        -- zacetek pogovora
                        begin dialog conversation @dialog_handle
                        from service [Metronik_Service]
                        to service 'Jinfo_Service'
                        on contract [JINFO_METRONIK_CONTRACT]
                        WITH ENCRYPTION = OFF;

                        -- Posiljanje XML-ja
                        send on conversation @dialog_handle
                        message type PrimkaKnjizena (@msg)

                        -- Flag da je message poslan
                        set @message_sent = 1
```

Sedaj, ko sporočilo pride do prejemnikove vrste, je na vrsti njihova procedura, da iz vrste prebere zapis in ga razčleni iz XML sporočila ter prepiše v njihovo tabelo, kjer so zapisi o dokumentu. S tem dejanjem je krog sklenjen in podatki so konsistentni.

2.3.8 Pretvorba zapisa iz tabele v XML dokument

Kreacija XML dokumenta je zelo preprosta. Sestavimo tipičen SELECT stavek in po potrebi povežemo z drugimi tabelami. V našem primeru ne uporabljamo naših ID-jev, temveč ID-je, ki smo jih dobili iz pošiljateljeve baze (Predhodno smo se tako dogovorili). V WHERE pogoju dodamo for xml raw ('PrimkaKnjizena'), root('Primka'), elemente ter morebitni pogoj, po katerem iščemo podatke. Procedura vrne XML dokument, ki vsebuje ROOT nivo ter pod nivoje dokumenta »primka«. XML dokument je lahko poljubno dolg.

```
CREATE function [custom_jgl].[sb_CreateMessage_PrimkaKnjizena]
(@dh_id int)
returns xml
as
begin
        declare @ret xml =
(SELECT  dd.external_id as Br_stavke,
                                        pl.s_lokacije as S_Lokacije,
                                        a.external_id as S_artikla,
                                        lot.material_cn as Kontrolni_Broj,
                                        lot.supplier_cn as Serijski_Broj,
                                        lot.best_before as VrijediDo,
                                        dd.quantity as Kolicina,
                                        isnull(iu1.ips_id,-10) as S_osobe
          FROM    warehouse.document_detail dd
          left join warehouse.document_header dh on
 dd.document_header_id = dh.id
          left join warehouse.lot lot on dd.lot_id = lot.id
          left join types.article a on dd.article_id = a.id
  left join custom_jgl.sb_ProcessedReceiptLines pl on
 header_id = dh.id and pl.br_stavke = dd.external_id
          left join dbo.sec_user_log_view su1 on
 dd.sec_user_log_id = su1.sec_user_log_id
          left join custom_jgl.ips_users iu1 on
 su1.login_name = iu1.login_name
         where   dd.document_header_id   =   @dh_id   for   xml   raw   ('PrimkaKnjizena'),
root('Primka'), elements)
return @ret
end
```

2.3.9 Mogoči problemi, do katerih lahko pride.

Pri vsaki informacijski rešitvi lahko pride do prekinitve. Pri prenosu podatkov lahko največkrat pride do prekinitve prenosa podatkov. Najbolj pogosta scenarija za prekinitev sta izklopljen prejemnikov strežnik (prekinitev električnega toka, ponovni zagon strežnika, izklopljena podatkovna baza …) in izguba podatkov med prenosom. Oba problema SB uspešno obvladuje. Če je strežnik izklopljen ali ni povezave do prejemnikovega strežnika, se sporočila nabirajo v vrsti pošiljatelja. Ko se povezava ponovno vzpostavi, se iz pošiljateljeve vrste pošlje v prejemnikovo vrsto, kjer se podatki normalno obdelajo. Tukaj ni nobene izgube podatkov. Podatki se samo s časovno zakasnitvijo prenesejo in obdelajo.

3 Sklepne ugotovitve

S tehnologijo SB-ja zelo enostavno prenašamo podatke med dvema strežnikoma. V podjetju smo preučili več tehnologij, vendar nobena ni ponujala toliko funkcionalnosti kot SB. Eden izmed glavnih dejavnikov pri izbiri tehnologije je bila garancija pri prenosu podatkov, saj morajo biti podatki v informatiki vedno pravilni, ažurni ter konsistentni.

Nadomestna tehnologija »Tehnologija vmesnih tabel« je sicer podobna, vendar dodane vrednosti rešitvi ne doda. Glavna slabost te tehnologije je obremenitev sistema, saj se glede na potrebo po točnosti podatkov opravilo lahko zelo pogosto izvaja. SB te slabosti nima, saj se zažene le takrat, ko so v tabeli podatki.

S tem projektom sem se naučil, kako prenašati podatke med dvema tujima podatkovnima bazama. Poleg te rešitve sem spoznal tudi drugo, zelo koristno lastnost SB-ja. Lahko ga uporabljamo tudi za obdelavo podatkov, da se izognemo ozkemu grlu ob povečanju vpisa podatkov v bazo (ob določenih intervalih dneva). To je zelo zanimiva funkcionalnost, ki jo SB ponuja.

Tehnologijo bom uporabljal še v prihodnosti, ker jo je zelo enostavno vpeljati v sistem. Je zelo varna in zanesljiva. Po potrebi lahko podatke tudi šifriramo, vendar to predstavlja dodatno znanje, ki pa ga še nisem osvojil.

4 Priloge

Primer sporočila, ki je bilo uspešno obdelano. Na sliki sta zapisa iz tabele Dobljeno iz sporočila iz IPS-a na strani 25.

	id	code	reference_document_code	reference_document_date	document_type_id	start_object_id	end_object_id	is_reversal	creation_date	posting_date	transfer_date	document_status_id	typ_plant_level_id	flow_reason_id	business_partner_id	to_act
1	12	Pri11-00003/11	R-374/2010	2010-12-29 00:00:00.000	4	127	3	0	2011-01-03 00:00:00.000	2011-01-03 08:33:44.273	2011-01-03 08:33:55.467	2	2	123	127	0

Slika 12 Glava dokumenta »Primka«

	id	document_header_id	step_number	start_object_id	end_object_id	lot_id	material_cn	supplier_cn	article_id	article_code	article_name	best_before	quantity	flow_reason_id	external_id	sec_user_log_id
1	948	12	1	127	3	909	A - 0002/11	1032/10	3235	ASUB096	UPUTA FOLACIN - (zbirna) RUS	2013-01-03 00:00:00.000	22700.000000	1	3	6132
2	949	12	2	127	3	910	A - 0003/11	1029/10	3242	ASUB098	UPUTA FOLACIN 30 tbl. - UKR	2013-01-03 00:00:00.000	5000.000000	1	4	6132
3	950	12	3	127	3	911	A - 0004/11	1121/10	6552	ASSO613	SL.K. MERALYS sprej 0,05% - HR	2013-11-03 00:00:00.000	22500.000000	1	5	6132
4	951	12	4	127	3	912	A - 0005/11	1123/10	6553	ASUO612	UPUTA MERALYS sprej (zbirna) - HR	2013-11-03 00:00:00.000	50000.000000	1	6	6132

Slika 13 Stavke dokumenta "Primka"

5 Kazalo slik

Slika 1: Pošiljanje sporočil....5
Slika 2: Pogoj sprožilca [1]....8
Slika 3: Konceptualni model - vmesna tabela....13
Slika 4: Kreacija opravila....14
Slika 5: Vpisovanje korakov....15
Slika 6: Časovni interval zagona....15
Slika 7: Konceptualni model Service Broker-ja....19
Slika 8: IPS informacijski sistem [2]....20
Slika 9: MePIS informacijski sistem [3]....21
Slika 10 Prikaz toka informacij iz IPS-a v MePIS....24
Slika 11 Prikaz toka informacij iz MePIS-a v IPS....27
Slika 12 Glava dokumenta »Primka«....37
Slika 13 Stavke dokumenta "Primka"....37

6 VIRI

[1] Aschenbrenner, K. (2008). *Pro SQL Server 2008 Service Broker*. Berkeley, California: Apress.

[2] IPS. Prevzeto 2011 iz http://www.jadraninfo.hr/ips_proizvodnja.asp

[3] Metronik. Prevzeto 2011 iz http://www.metronik.si/en/solutions/solutions-for-industry/informacijska-podpora-proizvodnim-procesom-s-sistemom-mepis

[4] Typical Uses of Service Broker. Prevzeto 2011 iz http://msdn.microsoft.com/en-us/library/ms166071.aspx