Sperasoft

Your game dev partner

SQL Server Service Broker

Scale out by Reliable Messaging

Agenda

- Scalability – what does Scale Out means?
- Sharding
- Problems of sharding
- How does Service Broker help?

Scalability

- Is the ability of a system to handle growing amount of work in a capable manner
- or its ability to be enlarged to accommodate that growth

http://en.wikipedia.org/wiki/Scalability

Scalability Methods

- Scale up – scale vertically
- Scale out – scale horizontally

Scaling Up

- means vertically
- add resources to a single node in a system
  - CPU
  - memory
  - storage
  - etc

Scaling Out

- means horizontally
- add more nodes to a system

- in databases that's called Sharding
- Microsoft used another term in SQL Azure - Federations

Sharding Frameworks

- Twitter Gizzard - https://github.com/twitter/gizzard
- written in Scala

Sharding

Sharding involves two technologies:

- Partitioning
- Replication

Partitioning

- Data is divided into chunks
- Stored across computers
- Each chunk is small enough that we can manipulate and query efficiently

User = “Denis”

User = “Eva”

Replication

- multiple copies of data are stored across several machines
- efficiently responds to tons of queries
- resilient to failures

User = “Eva”
User = “Denis”

User = “Eva”
User = “Denis”

Benefits of Sharding

Besides Scaling Out:

- Disaster Recovery strategies have You to make database backups
- Replicated shards are working copies

- Need a new backup? Replicate a new Shard!

Are there any Problems?

- Yes! Sharding is difficult!
- We have to think of how to partition our data correctly
- How to ensure that all copies of data are consistent?
- Write conflicts – no guarantee that operations will apply in order

Design Principle

- “Shared Nothing Architecture”
- no ongoing need to retain shared access between shards
- each Shard can live in a totally separate
  - instance
  - database server
  - data center
  - continent

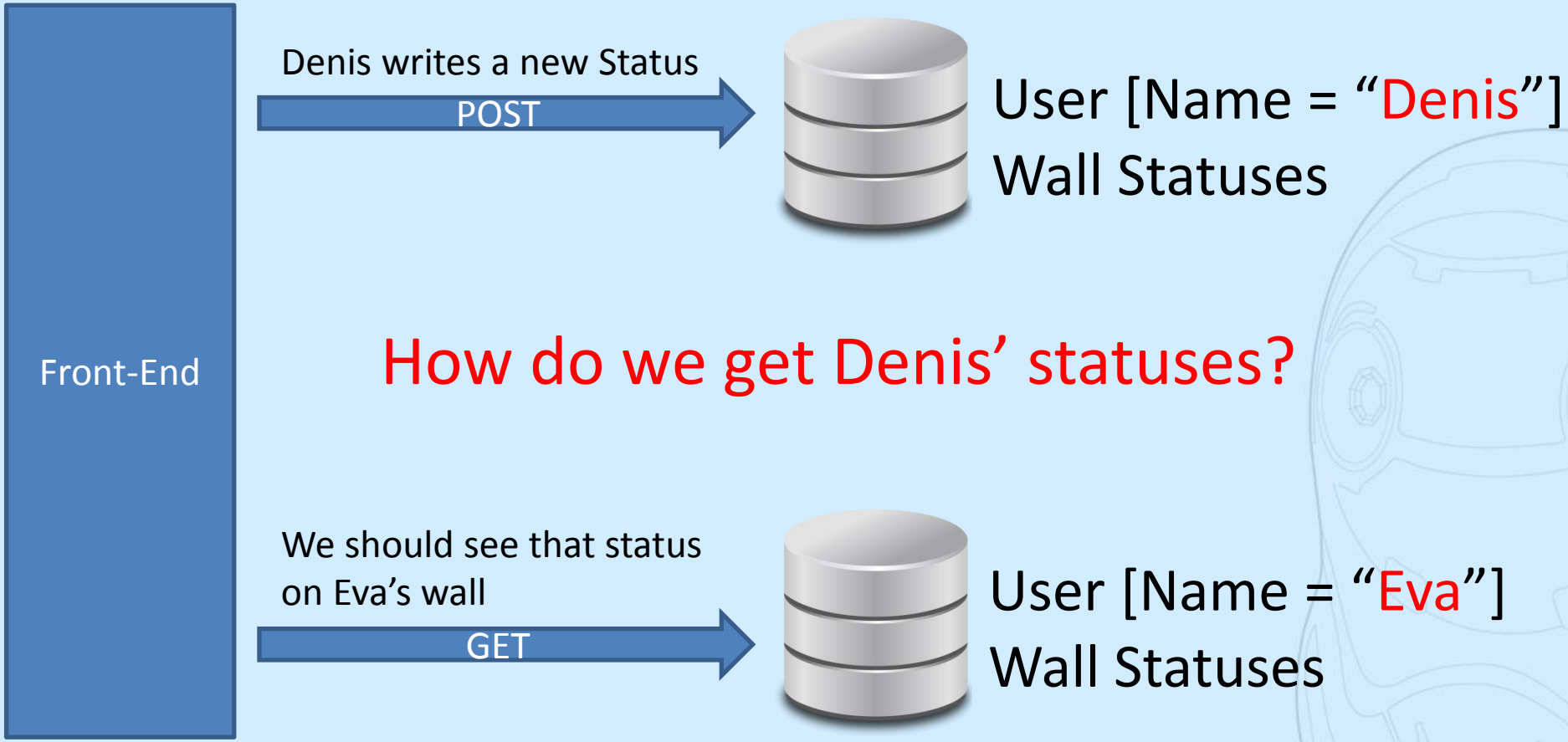
Partitioning Problem

- Orders table is partitioned by Region
- Orders references to Users
- Users is not partitioned, it is replicated
- How to maintain Users in a consistent state?

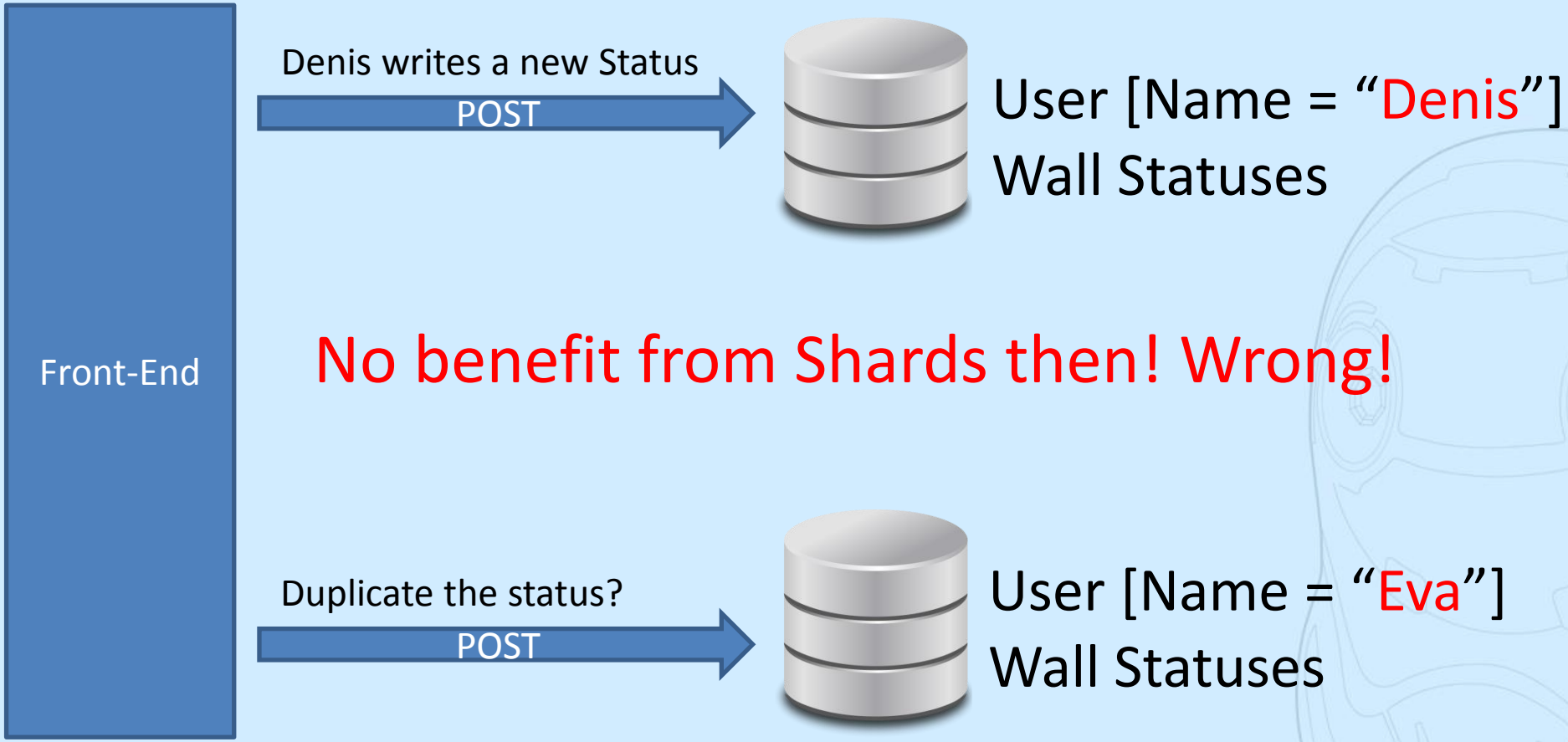
Orders [Region = “US”]
Users (not partitioned)

Orders [Region = “Russia”]
Users (not partitioned)

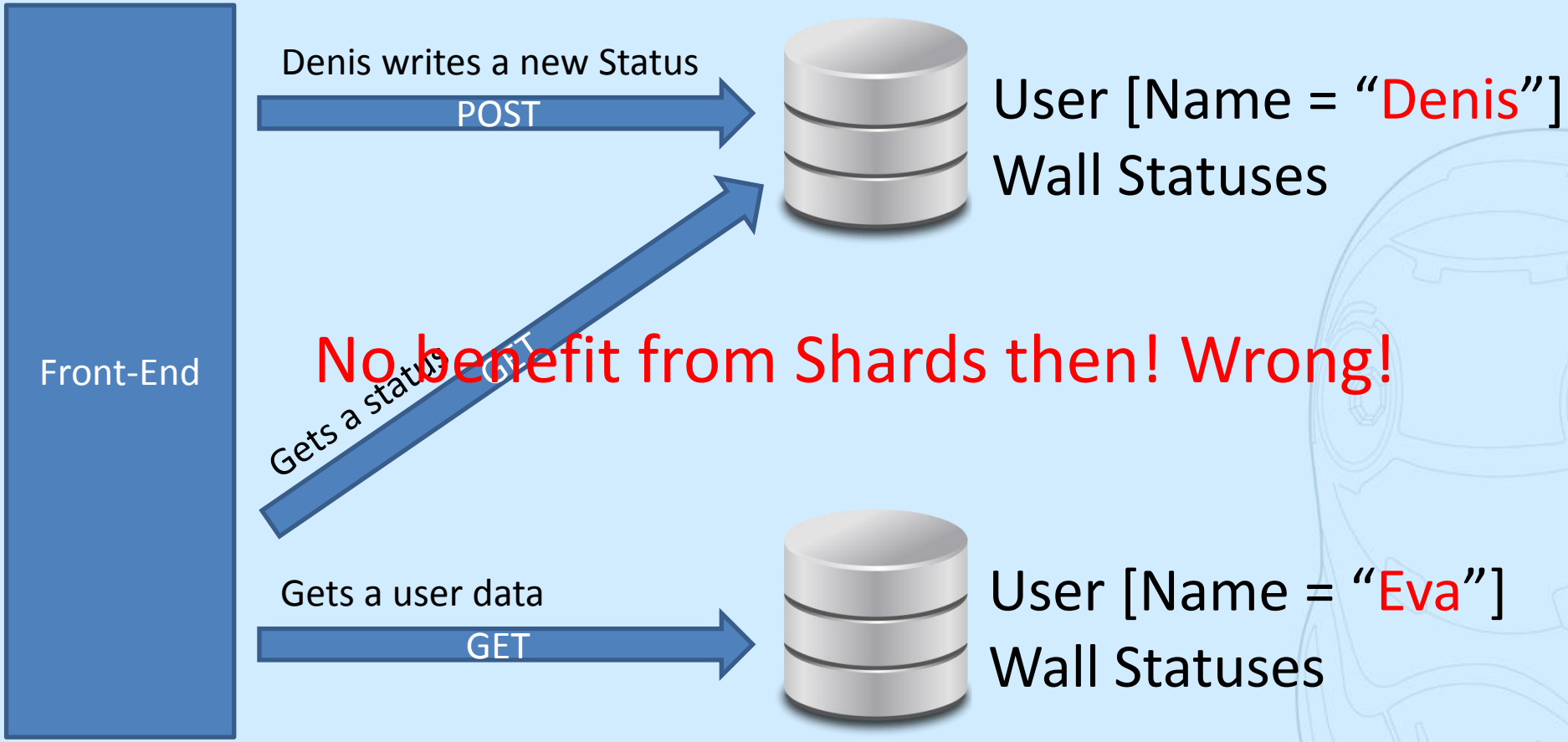
Facebook Sample



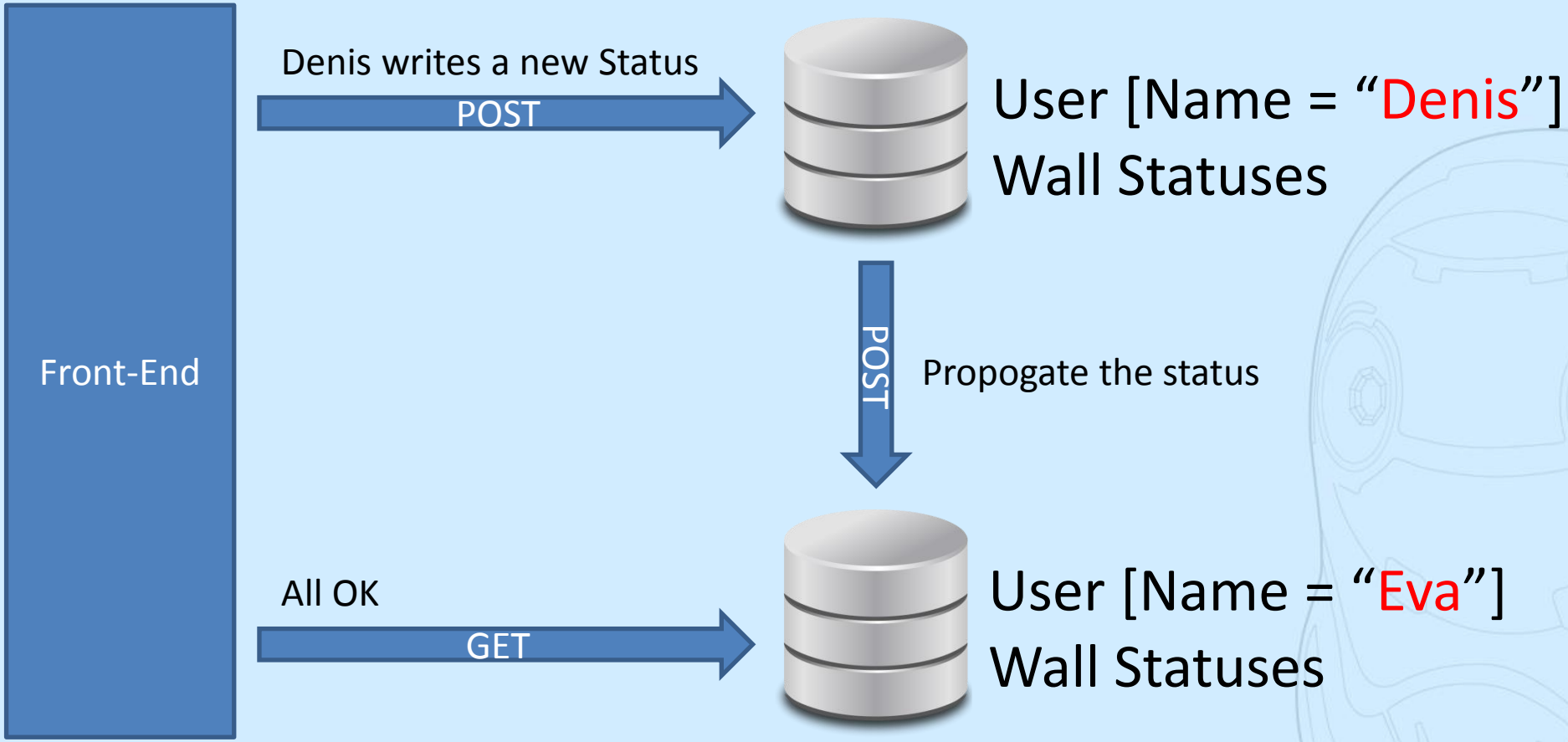
Possible Solution 1



Possible Solution 2



Real Solution



Possible Technologies

- custom application which reads from one Shard and writes to the others
- add all Shards as Linked servers, write stored procedures to write a new record to each remote server
- SQL Server Service Broker

Service Broker

- allows Internal and External processes to send and receive guaranteed asynchronous messages
- messages are sent to Queues in the same database, same instance, same server, another database, another instance, remote server

Implementations

the largest known implementation is in MySpace

- 440 SQL Server instances
- over 1,000 databases
- 1 Petabyte of data (1 million gigabytes)

Performance:

- 5,000 messages/second in Labs
- 18,000 messages/second in MySpace

Advantages

- Always Transactional
- Asynchronous
- Queued
- All messages are in XML format
- Routing
- Multicast messages in SQL Server 2012

Disadvantages

- No Administration Tools – not a lack - no at all
- Queue is a Table in database – index fragmentation when we cannot empty queue
- complicated T-SQL syntax and object model

Service Broker Concepts

- Message Types
- Contracts
- Queues
- Services
- Routes
- Activation
- Remote Service Bindings
- Dialog Conversation
- Conversation Groups

Service Broker Security

- Dialog Security – between two Services
  - encrypts messages in a dialog conversation
  - provides the remote authorization
- Transport Security
  - establishes an authenticated network connection between two Databases

SQL Server Configuration

- Enable Service Broker at Database level

```
ALTER DATABASE SampleDatabase
   SET ENABLE_BROKER;
GO
```

- Always enable Service Broker in MSDB system database in each SQL Server instance

- Enable SQL Server to communicate to another instance at Database level

```
ALTER DATABASE SampleDatabase
   SET TRUSTWORTHY ON;
GO
```

Message Types

- *Create Send Message type*

CREATE MESSAGE TYPE NewUserRequest

VALIDATION = WELL_FORMED_XML;

- *Create Receive Message Type*

CREATE MESSAGE TYPE NewUserResponse

VALIDATION = WELL_FORMED_XML;

Service Contracts

- create Contract between two services

```
CREATE CONTRACT NewUserContract (
   NewUserRequest    SENT BY INITIATOR,
   NewUserResponse SENT BY TARGET
);
```

Queues

```
CREATE QUEUE NewUserReceiveQueue
  WITH STATUS = ON,
  ACTIVATION (
    PROCEDURE_NAME = OnReceiveNewUser,
    MAX_QUEUE_READERS = 5,
    Execute AS 'dbuser‘
) ;

CREATE QUEUE NewUserSendQueue
  WITH STATUS = ON;
```

Services

- Service will be listening for messages in a queue and react only on those which apply to the spicified contract

```
CREATE SERVICE NewUserSendService
  ON QUEUE NewUserSendQueue (
    [NewUserContract]
);

CREATE SERVICE NewUserReceiveService
  ON QUEUE NewUserReceiveQueue (
    [NewUserContract]
);
```

Sending a Messgae

Sender:

- begins a transaction
- begins a dialog conversation from NewUserSendService to NewUserReceiveService
- sends one or more XML messages to that dialog
- commits a transaction

- Waits for a Reply message
- Processes Reply messages when arrived

Receiving a Message

Receiver (Target):

- when a new message is delivered to Receiver's NewUserReceiveQueue the OnReceiveNewUser stored procedure is activated by Service Broker
- Stored procedure begins a transaction
- receives messages out of the NewUserReceiveQueue queue
- sends Reply messages
- ends a dialog conversation
- Do what you want with that message
- commits a transaction