SQLintersection

Going Asynchronous with Service Broker

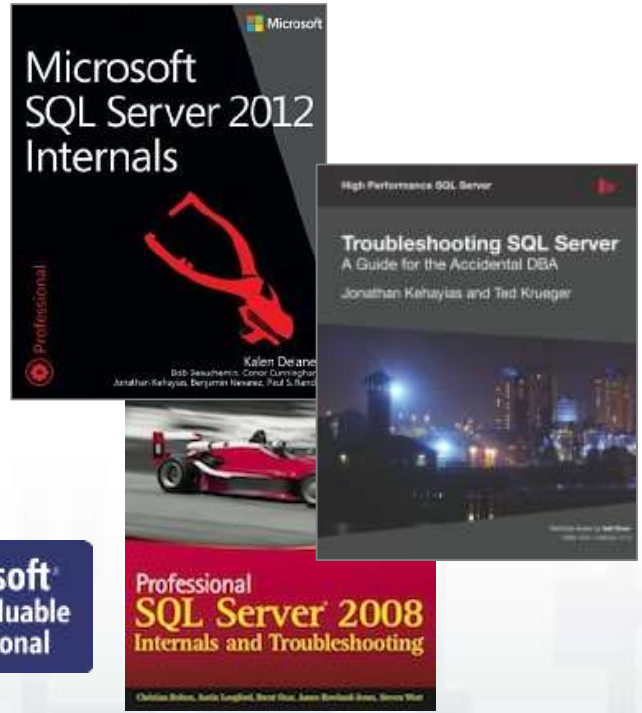
Jonathan Kehayias
Principal Consultant
SQLskills.com

SQL *intersection*

Jonathan Kehayias

- **Consultant/Trainer/Speaker**
- **Principal Consultant, SQLskills.com**
  - Email: Jonathan@SQLskills.com
  - Blog: http://www.SQLskills.com/blogs/jonathan
  - Twitter: @SQLPoolBoy
- **SQL Server MVP since October 2008**
- **Microsoft Certified Master: SQL Server 2008**

SQLskills
immerse yourself in sql server



Microsoft CERTIFIED Master SQL Server 2008

MVP Microsoft Most Valuable Professional

SQL intersection

Reminder: Intersect with Speakers and Attendees

- **Tweet tips and tricks that you learn and follow tweets posted by your peers!**
  - Follow: #SQLintersection and/or #DEVintersection
- **Join us – Wednesday Evening – for SQLafterDark**
  - Doors open at 7:00 pm
  - Trivia game starts at 7:30 pm

    Winning team receives something fun!
  - Raffle at the end of the night

    Lots of great items to win including a seat in a SQLskills Immersion Event!
  - The first round of drinks is sponsored by SentryOne and SQLskills

SQLafterDark

SQL intersection

SQLskills immerse yourself in sql server & SentryOne™

Overview

- Service Broker usage scenarios
- Advantages of Service Broker
- Basic Service Broker architecture
- Conversation architectures
- Understanding activation

SQL intersection

Usage Scenarios (1)

- **Centralized data collection**
  - Multiple sites process transactions locally and use Service Broker to send information to a central office
  - Asynchronous message delivery allows local sales transactions to continue even if a site loses connectivity to the central office
- **Asynchronous trigger execution**
  - Complex logic in DML triggers affect OLTP performance
  - Queuing a message requesting work from a Service Broker service to offload the complex logic into a separate transaction asynchronously

SQL intersection

Usage Scenarios (2)

- **Reliable processing of queries**
  - Single transaction is used to read a message, run a query, and return results to the initiating service
  - During a service failure the transaction would roll back, return the message to the queue, and restart processing after recovery
- **Distributed server-side processing**
  - Online ticketing application processes orders using Service Broker to exchange data between payment processing, CRM, and ticket printing applications

SQL intersection

Advantages of Service Broker

- **Database integration**
  - Eliminates the overhead of a separate distributed transaction coordinator
  - The message store and database are maintained at the same transactional point-in-time, eliminating the need for reconciliation processes when one of the two has to be restored
- **Ordering of messages**
  - Messages sent within the Service Broker architecture are received once and only once in the order in which the messages were sent

SQL intersection

Advantages of Service Broker (2)

- **Message locking of related messages**
  - Service Broker prevents out of order processing of message within the same conversation group by locking the conversation group during processing
  - Multiple threads can process different conversation groups concurrently increasing scalability and performance

SQL intersection

Advantages of Service Broker (3)

- **Loose coupling of applications**
  - The initiating application and target application are loosely coupled, allowing the initiating application to continue processing after sending a message to the target application
  - Loose coupling provides scheduling flexibility, allowing parallel processing
  - Background processing of requests can mitigate high load times against source and target servers through queuing requests for processing

SQL *intersection*

Advantages of Service Broker (4)

- **Automatic activation**
  - Activation allows Service Broker to automatically scale to match the volume of messages arriving in a given service queue
  - Allows programs running inside the database as well as external to the database to take advantage of queue activation
  - Automatically starts a new queue reader when there is work to be performed

SQL intersection

Broker and SQL Server Features

- Event Notifications – watch for Trace/DDL events and respond automatically when they occur
- Query Notifications – monitor for changes in results, and leveraged for external queue activation of Service Broker
- Database mail – implemented as an external activated application that monitors for messages being queued in msdb
- WMI Events – piggy back on Event Notifications to allow integration with Windows Management Instrumentation

SQL intersection

Basic Service Broker Architecture

- Single database implementation for asynchronous processing
- Enable the database for service broker
- Create request/reply message types
- Create contract specifying request messages are sent by the initiator and reply messages are sent by the target
- Create a target/initiator queues/services

SQL intersection

Basic Service Broker Architecture (2)

- Create the target service specifying the contract to limit conversations
- Create the initiator service with no contract
- Intra-database Service Broker requires 4 types of objects:
  - Message types, contracts, queues, and services

SQL intersection

Service Broker Components

- **Message types**
  - Define the name and type of data that will be in a message
  - SQL Server can validate that a message contains valid XML, XML conforming to a specific schema, or no data at all
  - SQL Server can also not validate message data, accepting any data for the message including binary data, XML, or empty messages
  - System message types exist for handling errors and dialog status Default message type used when a message type is not specified

SQL intersection

Service Broker Components (2)

- **Contracts**
  - Define which message types will be used for specific tasks
    - Tasks are defined in Service Broker as initiator (or sender) and target (or receiver)
  - Agreement between services about which message types each of the services will send for a given task
  - Must contain a message type sent by the initiator, otherwise there is no way for the initiator to begin a conversation using the contract

SQL intersection

Service Broker Components (3)

- **Queues**
  - Store messages in the database for processing by applications
  - When messages arrive at a service they are inserted into the queue associated with the service
  - Each message is a row in the queue containing:
    - Contents of the message, message type and validation performed
    - Service and contract targeted by the message
    - The conversation the message is associated with
    - Internal information to the queue

Service Broker Components (4)

- **Queues (2)**
  - Applications RECEIVE messages from the queue to get the message for processing within a transaction
    - All messages for a conversation or a conversation group can be processed at once
    - Queues return messages from each conversation in the order the messages were sent
  - May use a stored procedure for internal activation (automated processing)

SQL intersection

Service Broker Components (5)

- **Services**
  - Specifies the name of a specific business operation or task implemented in Service Broker
  - Service names are used to:
    - Deliver messages to the correct queue inside of a database
    - Route messages from target to initiator, and back
    - Determine security requirements for new conversations
    - Enforce appropriate contracts for conversations
  - Bound to a specific queue to store incoming messages

SQL intersection

Demo

- Basic Service Broker architecture

SQL intersection

Conversation Architecture

- **Service Broker applications communicate using a dialog conversation or persisted stream of messages back-and-forth between two services**
  - Dialogs provide exactly-once-in-order (EOIO) message delivery using a conversation identifier and sequence numbers
- **Dialogs have two participants:**
  - Initiator – starts the conversation
  - Target – accepts a conversation started by an initiator
  - Messages provide the information exchanged

SQL intersection

Processing Messages

- **The RECEIVE DML statement is used to process and remove messages**
  - Receive a single message into variables with TOP 1 (slowest method)
  - Receive all messages for a single conversation into table variable (faster method)
  - Receive all messages for a conversation group into table variable (faster method)
- **Messages are always received in sequence order, per conversation**

SQL intersection

Processing Messages (2)

- **The RECEIVE statement will lock the conversation using a Service Broker-specific internal lock in SQL Server**
  - This will be a bottleneck if a single conversation is used for all messages
- **Transactions should be used when processing messages, with error handling, to prevent poison messages from occurring**

Understanding Dialog Conversations

- **Each dialog conversation has a unique conversation handle associated with the conversation**
  - The conversation handle is created on the initiator by the BEGIN DIALOG CONVERSATION command
  - The conversation handle is retrieved by the target when it processes the message(s) with RECEIVE from the queue
  - The target uses the conversation handle to send a response message(s) back to the initiator and ends the conversation with END CONVERSATION

SQL *intersection*

Understanding Dialog Conversations (2)

- **Certain interactions may require a back-and-forth series of messages on the same conversation with different message types defined by the contract**
  - The initiator should end the conversation when it processes the final message from the target using END CONVERSATION
  - The target should end the conversation based on receipt of a specific message type to avoid orphaning the conversation
- **“Fire and forget” is good for the military but not Service Broker**
  http://bit.ly/SSB_FireForget

SQL intersection

Basic Conversation Example
Sending a Request Message

- **Begin a conversation at the initiator specifying (BEGIN DIALOG):**
  - The sending (initiator) service name
  - The receiving (target) service name
  - The contract name for the conversation
- **Sending a message on the conversation (SEND ON CONVERSATION) specifying the message type and the message body to be sent**

SQL intersection

Basic Conversation Example
Processing Request and Replying

- Receiving the message from the target queue (RECEIVE)
- Sending a response message from target to initiator (SEND ON CONVERSATION) specifying the message type and the message body to be sent

SQL intersection

Basic Conversation Example
Processing Reply and Ending Dialog

- **Receiving the response message on the initiator (RECEIVE)**
- **Ending the conversation on the initiator (END CONVERSATION)**
  - Sends an EndDialog message to the target to cleanup the conversation handle
  - Ending the dialog on the target (END CONVERSATION)

SQL intersection

Demo

- Basic intra-database conversation

SQL intersection

Dialog Conversation Internals

- Reliable delivery through receipt-acknowledgement from the target service
- Outgoing messages stored in the transmission queue until acknowledged by the target service
- If a service is unavailable, messages remain in the transmission queue and retries every minute until the conversation is ended, or the lifetime of the conversation expires,

SQL intersection

Activation

- **Allows automatic processing of messages in a queue**
- **For applications inside of the database a stored procedure can be used to process the queued messages**
  - Referred to as internal activation
  - Stored procedure is associated with a queue and starts automatically
  - Must have an owner and use EXECUTE AS for security context
  - Certificate signing should be used for cross-database access

SQL intersection

Activation (2)

- **Service Broker also supports activation outside of the database through an application or service monitoring the queue**
  - This can be written explicitly in the application or service
  - The External Activation feature can alternatively be configured to execute the program or service

SQL intersection

Demo

- Activation demo

Review

- Service Broker usage scenarios
- Advantages of Service Broker
- Basic Service Broker architecture
- Conversation architectures
- Understanding activation

SQL intersection

Questions?

SQLintersection

Don’t forget to complete an online evaluation!

Going Asynchronous with Service Broker

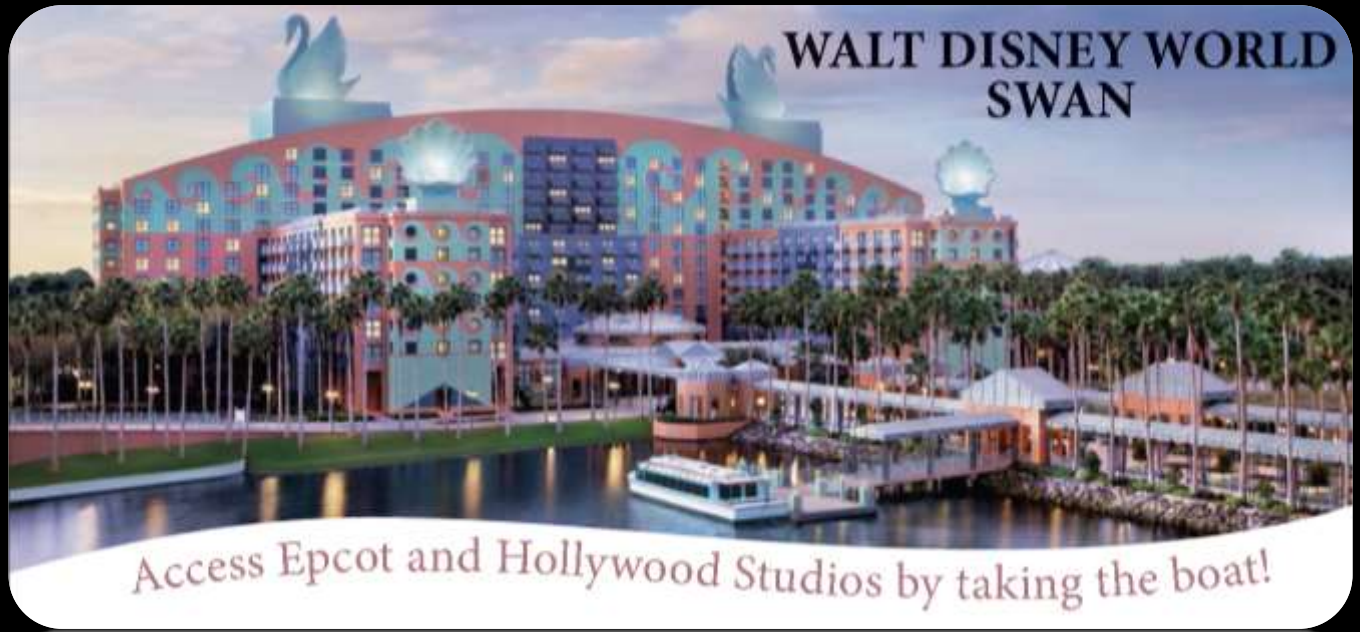
Your evaluation helps organizers build better conferences and helps speakers improve their sessions.

SQL *intersection*

Thank you!

Save the Date!

www.SQLintersection.com



2018

Mar 25-28

We're back in Orlando!

SQL intersection

Leave the every day behind and enter a world of wonder and enchantment at the Walt Disney World® Resort. Located in the heart of the most magical place on earth, the Walt Disney World Swan and Dolphin Resort provides a truly extraordinary backdrop for our event! Beautiful tropical landscaping, tranquil waterways, and classic art and architecture work together to create a stunning landmark!