HERZLICH WILLKOMMEN

Ein fast vergessenes Feature

Oracle Advanced Queueing

DOAG 2017 in Nürnberg

Axel vom Stein

axel.vomstein@bss-bohnenberg.de

Unternehmen – Fakten …

Hauptsitz:	**Solingen**
Standorte:	**Ilmenau (Deutschland), Oedelem (Belgien), Kattowitz (Polen), Wels (Österreich)**
Geschäftsführer:	**Martin Gräb, Vasilios Dossis**
Gegründet:	**1991**
seit 2006:	**Unternehmen der ROFA – Gruppe**
Rechtsform:	**GmbH**
Mitarbeiter:	**ca. 140**

Unternehmen – BSS befasst sich mit …

- Entwurf und Feinplanung von intralogistischen Systemen und Anlagen
- schlüsselfertiger Abwicklung als GU
- Flowpicker (Hochleistungskommissionierung)
- Hochregal und Kleinteilelagern
- Lagerverwaltungssystemen
- Materialflussrechnern
- fahrerlosen Transportsystemen
- Pick-by-Light / Pick-by-Voice
- Staplerleitsystemen
- SPS – Technik
- Visualisierungen
- etc.

Über mich …

- **Technischer Projektleiter (bei BSS seit 2001)**

- **Oracle:**
  - **seit 2001**
  - **zunächst als Entwickler (PL/SQL, C#, Delphi)**
  - **dann mehr und mehr Richtung DBA**

- **Schwerpunkte/Fables:**
  - **Oracle auf Windows**
  - **Oracle Standard Edition**
  - **Sicherheits/HA - Konzepte**
  - **Migrationen**
  - **Standardisierung**
  - **alles was sich automatisieren lässt**

Agenda

1.) Hintergrund

- **Historie AQ**
- **Was ist AQ?**

2.) Wie geht’s?

3.) Zusammenfassung

Hintergrund: Historie AQ

- **erstmals mit Oracle 8.0, Feature der EE (1998)**
- **seit Oracle 9.2 auch in der SE (2002)**
- **seit Oracle 10g sogar in der XE (2004)**
- **Oracle nutzt es selbst:**
  - **Data Pump, Scheduler**
  - **Database Change Notification (PL/SQL, C# ODP.Net, JDBC)**
  - **Streams, Golden Gate**
- **Oracle Dokumentation:**
  - **Streams deprecated 12.1 → use Golden Gate**
  - **AQ is independed of Streams and continues to be enhanced**

Hintergrund: Was ist AQ?

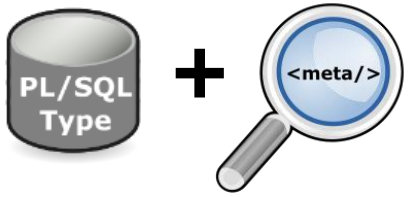
- **Datenbank integriertes Message System**
- **Asynchrone Interprozesskommunikation mit (verteilten) Anwendungen**
- **Grundprinzip: Ereignis-orientiert statt Pollen**
- **langlaufende Prozesse im Hintergrund, GUI steht sofort zur Verfügung**
- **lang erprobte und somit ausgereifte Technologie**
- **reine DB – Lösung, d.h.**
  - **→ Datenintegrität gegeben**
  - **→ Sicher, Skalierbar, Verlässlich**
  - **→ Desaster (Recovery)**
  - **→ Hochverfügbar**

Hintergrund: Was ist AQ? (Definitionen)

- **Queue Typ: Definition der Anwendungsdaten**

- **Metadaten: Eigenschaften, z.B. Delay, Priority, etc.**

- **Message: Queue Typ + Metadaten**



- **Payload: Inhalt einer Message**

- **Queue:**

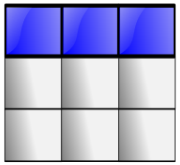
  Warteschlange, erstellen und abrufen von Messages

- **Exception Queue:**

  auch AQ$-Queue, Messages, die nicht verarbeitet wurden

- **Queue Table:**

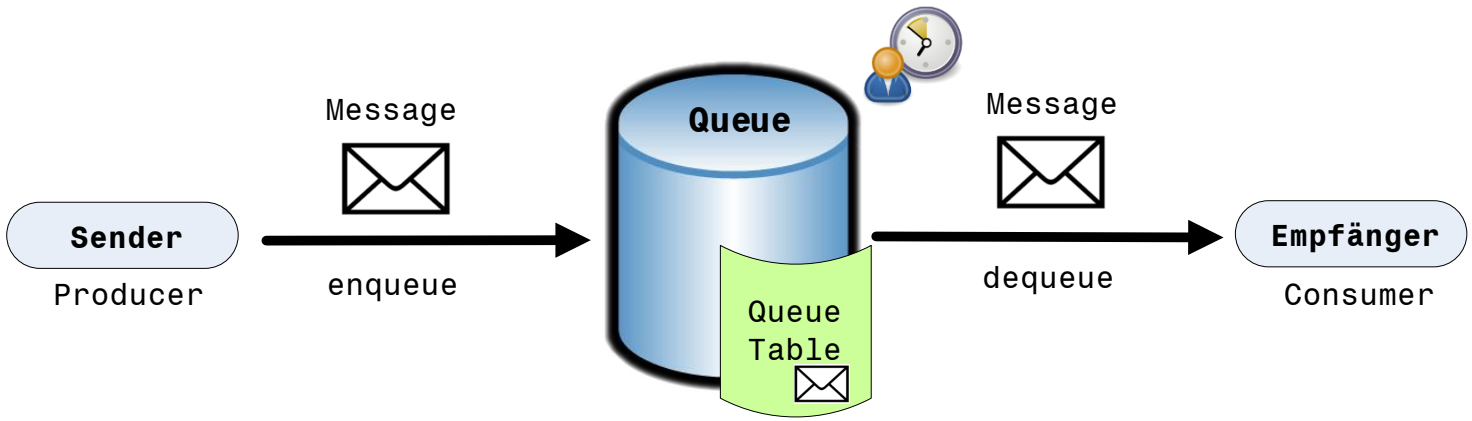
  Ablage aller Messages basierend auf. Queue Typ



- **Time Manager:**

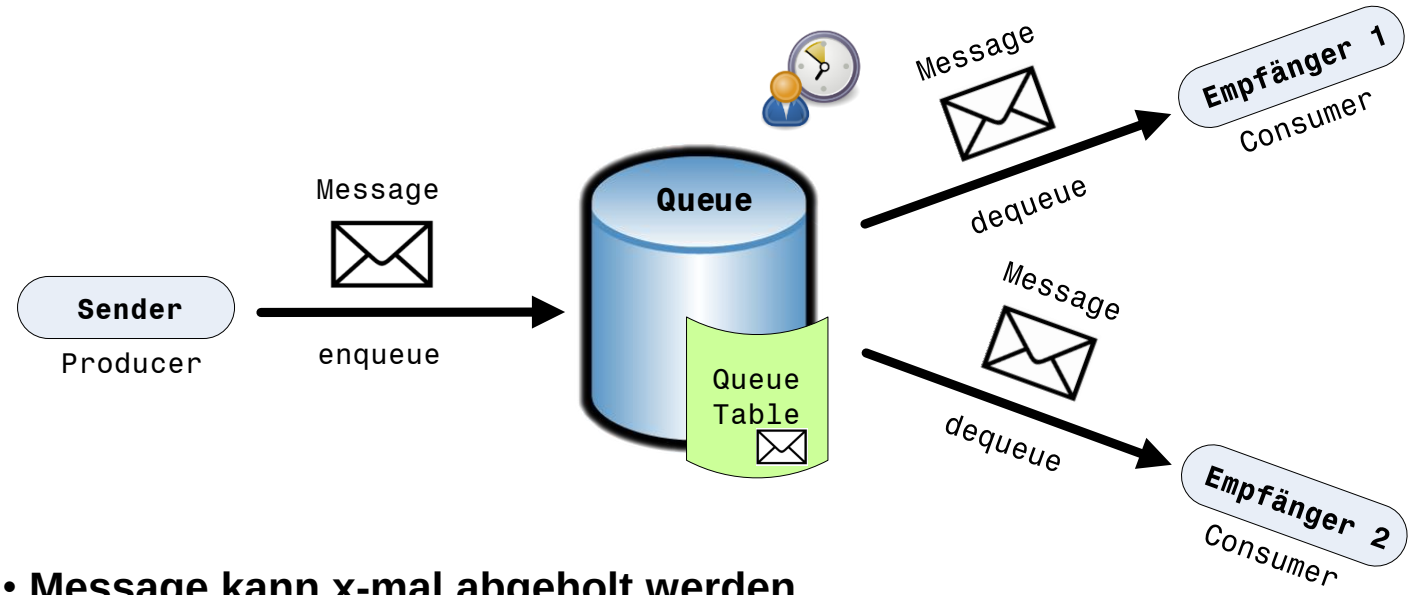
  wann wird was mit einer Nachricht gemacht

- P2P (Point to Point) oder Single Consumer Queue



- Message kann genau einmal abgeholt werden

- **PS (Publish – Subscribe) oder Multi Consumer Queue**



- **Message kann x-mal abgeholt werden**
- **Jeder Empfänger erhält die gleiche Nachricht**
- **Message verbleibt in der Queue, bis alle registrierten Abonnenten abgeholt haben oder das eingestellte Timeout abgelaufen ist**

Agenda

1.) Hintergrund

2.) Wie geht’s?

- Elf Schritte - Nix nach Nix
- Message Management
- Stolpersteine
- Sonstiges

3.) Zusammenfassung

Hauptbestandteil folgende PL/SQL Packages:

1.) SYS.DBMS_AQ (enqueue, dequeue)

2.) SYS.DBMS_AQADM (Administration von Queues)

- **Berechtigungen (eigene Queue):**

  create, drop, manage own queues (execute on DBMS_AQADM)

  enqueue, dequeue own queues (execute on DBMS_AQ)

- **Administrative Berechtigungen (alle Queues):**

  create, drop, manage any queues (+ AQ_ADMINISTRATOR_ROLE)

- **Berechtigungen (dequeue, enqueue auf fremde Queue):**

```
+ SYS.DBMS_AQADM.GRANT_QUEUE_PRIVILEGE
     (
      PRIVILEGE  => 'DEQUEUE',  -- ALL, ENQUEUE
      QUEUE_NAME => QueueName,
      GRANTEE    => QueueDequeueGrantee
     );
```

- **Systemprivilegien**

  - ENQUEUE_ANY_QUEUE
  - DEQUEUE_ANY_QUEUE
  - MANAGE_ANY_QUEUE
  - …

1.) Vorbereitungen

```
CREATE USER DOAG IDENTIFIED BY doag;
ALTER  USER DOAG QUOTA 400M ON "USERS";

GRANT CONNECT                    TO DOAG;

GRANT CREATE TYPE                TO DOAG;

GRANT EXECUTE ON SYS.DBMS_AQ     TO DOAG;
GRANT EXECUTE ON SYS.DBMS_AQADM  TO DOAG;
```

2.) Queue-Typ anlegen

```
CREATE TYPE DOAG.Message_typ AS OBJECT
   (
    Subject  VARCHAR2(30),
    Text     VARCHAR2(80)
   );
```

3.) Queue-Tabelle anlegen

```
BEGIN
  SYS.DBMS_AQADM.CREATE_QUEUE_TABLE
     (
      Queue_Table        => 'DOAG.MESSAGE_QTABLE',
      Queue_Payload_Type => 'DOAG.MESSAGE_TYP'
     );
END;
/
```

Select auf DBA_QUEUE_TABLES:

OWNER	DOAG
QUEUE_TABLE	MESSAGE_QTABLE
OBJECT_TYPE	DOAG.MESSAGE_TYP
SORT_ORDER	ENQUEUE_TIME
RECIPIENTS	SINGLE
MESSAGE_GROUPING	NONE
COMPATIBLE	10.0.0

Select auf DBA_QUEUES:

OWNER	DOAG
NAME	AQ$_MESSAGE_QTABLE_E
QUEUE_TABLE	MESSAGE_QTABLE
QUEUE_TYPE	EXCEPTION_QUEUE
MAX_RETRIES	0
RETRY_DELAY	0
ENQUEUE_ENABLED	NO
DEQUEUE_ENABLED	NO
RETENTION	0

Es entstehen automatisch neue Views:

- DOAG.AQ$_Message_QTable_F
- DOAG.AQ$Message_QTable
- Und weitere bei Multi-Consumer-Queues

4.) Queue anlegen

```
BEGIN

  SYS.DBMS_AQADM.CREATE_QUEUE
     (
      Queue_Name  => 'MESSAGE_QUEUE',
      Queue_Table => 'DOAG.MESSAGE_QTABLE'
     );

END;
/
```

5.) Queue starten

```
BEGIN
  SYS.DBMS_AQADM.START_QUEUE
     (
      Queue_Name => 'DOAG.MESSAGE_QUEUE'
     );
END;
/
```

DBA_QUEUES:

OWNER	DOAG
NAME	MESSAGE_QUEUE
QUEUE_TABLE	MESSAGE_QTABLE
QUEUE_TYPE	NORMAL_QUEUE
MAX_RETRIES	5
RETRY_DELAY	0
ENQUEUE_ENABLED	YES
DEQUEUE_ENABLED	YES
RETENTION	0

6.) Queue-Nachricht einstellen

```
DECLARE
   Enqueue_Options     SYS.DBMS_AQ.Enqueue_Options_t;
   Message_Properties  SYS.DBMS_AQ.Message_Properties_t;
   Message_Handle      RAW(16);
   Message             DOAG.Message_typ;

BEGIN

   Message := Message_typ ( 'Erste Nachricht',
                            'Eigentlich ganz einfach!' );
   SYS.DBMS_AQ.Enqueue
      (
       Queue_Name         => 'DOAG.MESSAGE_QUEUE',
       Enqueue_Options    => Enqueue_Options,
       Message_properties => Message_Properties,
       Payload            => Message,
       Msgid              => Message_Handle
      );

   COMMIT;
END;
/
```

7.) Queue-Nachricht abholen

```
DECLARE
   Dequeue_Options     SYS.DBMS_AQ.Dequeue_Options_t;
   Message_Properties  SYS.DBMS_AQ.Message_Properties_t;
   Message_Handle      RAW(16);
   Message             DOAG.Message_typ;

BEGIN

   SYS.DBMS_AQ.DEQUEUE
      (
       Queue_Name          => 'DOAG.MESSAGE_QUEUE',
       Dequeue_Options     => Dequeue_Options,
       Message_Properties  => Message_Properties,
       Payload             => Message,
       Msgid               => Message_Handle
      );

   DBMS_OUTPUT.PUT_LINE ('Message: ' || Message.Subject || ': ' || Message.Text );
   COMMIT;
END;
/
```

SELECT auf DOAG.MESSAGE_QTABLE (vor erstem dequeue)

Q_NAME	MESSAGE_QUEUE
MSGID	F3809A821258455AB1E9D4EC10F48A0F
PRIORITY	1
STATE	0
DELAY	
ENQ_TIME	02.11.17 19:27:28,278000
ENQ_UID	DOAG
RETRY_COUNT	0
EXCEPTION_QUEUE	
USER_DATA.SUBJECT	Erste Nachricht
USER_DATA.TEXT	Eigentlich ganz einfach!

SELECT auf DOAG.MESSAGE_QTABLE (6x dequeue mit ROLLBACK)

Q_NAME	AQ$_MESSAGE_QTABLE_E
MSGID	F3809A821258455AB1E9D4EC10F48A0F
PRIORITY	1
STATE	3
DELAY	
ENQ_TIME	02.11.17 19:27:28,278000
ENQ_UID	DOAG
RETRY_COUNT	5
EXCEPTION_QUEUE	MESSAGE_QUEUE
USER_DATA.SUBJECT	Erste Nachricht
USER_DATA.TEXT	Eigentlich ganz einfach!

8.) Queue stoppen

```
BEGIN
  SYS.DBMS_AQADM.STOP_QUEUE
     (
      Queue_Name => 'MESSAGE_QUEUE'
     );
END;
/
```

9.) Queue entfernen

```
BEGIN
  SYS.DBMS_AQADM.DROP_QUEUE
     (
      Queue_Name => 'DOAG.MESSAGE_QUEUE'
     );
END;
/
```

10.) Queue-Tabelle entfernen

```
BEGIN
  SYS.DBMS_AQADM.DROP_QUEUE_TABLE
     (
      Queue_Table => 'DOAG.MESSAGE_QTABLE'
     );
END;
/
```

Views werden automatisch mit entfernt.

11.) Queue-Typ entfernen

```
DROP TYPE DOAG.Message_typ;
```

- **AQ bietet umfangreiche Möglichkeiten des MM:**

- **sort_key (Reihenfolge, in der Einträge aus der Queue geholt werden):**

  - **ENQ_TIME (default)**
  - **ENQ_TIME, PRIORITY**
  - **PRIORITY**
  - **PRIORITY, ENQ_TIME**
  - **COMMIT_TIME**
  - **PRIORITY, COMMIT_TIME**

Achtung: **Nach Anlegen der Queue-Table nicht änderbar!**

- **max_retries:**

  Anzahl Wiederholversuche nach Rollback

- **retry_delay:**

  Zeit zwischen erneutem Versuch nach Rollback, default=0

- **retention_time:**

  Verbleib der Nachricht in der Queue nach Abarbeitung

  default=0, INFINITE, Zeit in Sekunden

- **message_grouping:**

  Parent/Child – Nachrichten

  abholbar nur von einem User zur gleichen Zeit

- **delay:**

  Zeit in Sekunden, ab wann eine Nachricht „scharf“ geschaltet wird

  NO_DELAY (default)

- **priority:**

  Priorität einer Nachricht (ganzzahlig, je niedriger desto höher die Priorität)

- **expiration:**

  Verfallsdatum einer Nachricht (NEVER oder in Sekunden),

  falls nicht abgeholt, wird die Nachricht in die Exception-Queue verschoben

- **correlation:**

  Identifier beim Einstellen einer Nachricht

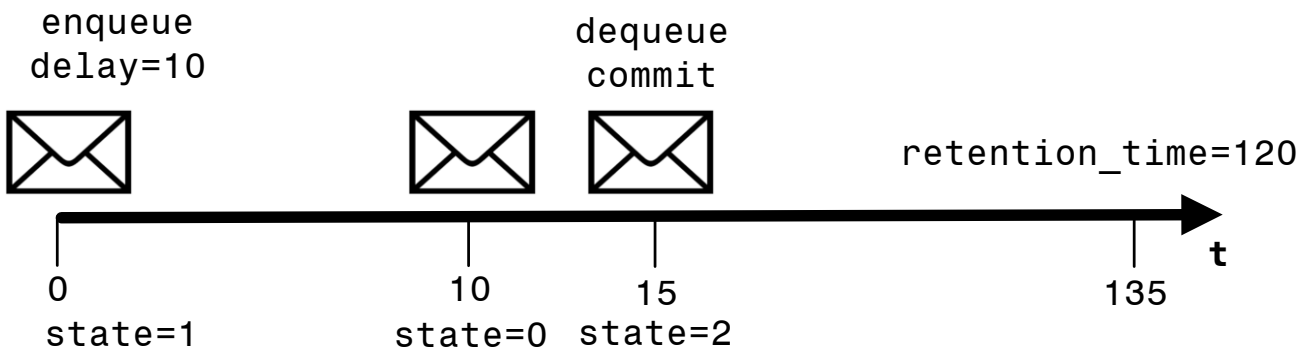
- **state (in Queue – Tabelle)**

  0 – READY

  1 – WAITING

  2 – PROCESSED

  3 – EXPIRED



- **Expired Queue-Eintrag wiederbeleben:**

  Update: retry_count=0, state=0, queue_name=MESSAGE_QUEUE

- **Shared Sessions in Kombination Queue mit WAIT – Attribut:**

  Zeit die gewartet wird, wenn keine Message für dequeue ansteht,

  FOREVER (default), NO_WAIT oder WAIT in Sekunden

  → ggf. wird der Shared Server Prozess blockiert, d.h. DEDICATED

- **direkte DDL statements auf Queue Tabellen:**

  nicht supported (z.B. Shrink führt zu defekter Queue)

- **Payloads mit LOB‘s:**

  User braucht Rechte auf der Queue Tabelle für enqueue/dequeue

  - `SELECT`
  - `INSERT`
  - `UPDATE`

- **Export / Import ist möglich, aber:**

1.) `FROMUSER`, `TOUSER` Klausel ist nicht supported

2.) `IGNORE` Parameter (Obj Create) auf `NO` setzen beim Import von Queue Tables

3.) `TABLE_EXIST_ACTION`: `TRUNCATE` und `APPEND` mode sind nicht supported

Problem bei 2.) u. 3.) sind ggf. doppelte Queue-Einträge,

(sofern die Queue-Definitionen überhaupt kompatibel sind)

Programmatic Interfaces for Oracle AQ

PL/SQL	DBMS_AQ, DBMS_AQADM
C	Oracle Call Interface (OCI)
C#	ODP.NET
Java	Oracle.JMS package using JDBC
AQ XML servlet	Internet Data Acces Presentation
…	

Durchführung: Sonstiges

- **Interessante Views:**
  - **dba_queues**
  - **dba_queue_tables**
  - **dba_queue_subcribers**
  - **dba_queue_publishers**
  - **v$aq (waiting, ready, expired, …)**
  - **v$aq1 (total_consumers)**
  - **…**

- **Prozesse:**
  - **Queue Monitor Coordinator (QMNC)**
    spawnt QXXX Prozesse abhängig von Systemlast
  - **Jnnn Prozesse**
    automatisch gestartet von Database – Scheduler,
    benötigt für Propagation + Notfication Jobs

- **init.ora – Parameter:**
  - **AQ_TM_PROCESSES (=0 → disable QMNC, nicht gesetzt → auto)**
  - **JOB_QUEUE_PROCESSES (DBMS_JOBS bedienen sich ebenfalls)**

Agenda

1.) Hintergrund

2.) Wie geht’s?

3.) Zusammenfassung

- Fazit
- Quellen

Zusammenfassung: Fazit

- **AQ ist sehr ausgereift**
- **AQ ist gut dokumentiert**
- **AQ Einstieg ist leicht**
- **AQ ist in die DB integriert, keine Third-Party Software notwendig**
- **Administration, Backup & Recovery out of the box**
- **Messages in vielen Ausprägungen selbst definierbar (auch LOB‘s)**
- **Abholen und Verarbeiten zu beliebiger Zeit**
- **AQ kostet nix**

Und es gibt noch einiges zu entdecken …

- **RAC (primary, secondary, owner – instance)**
- **Sharded Queues (12c)**
- **Transformation (Umwandlung von Datentypen)**
- **Propagation (Weiterleitung auf eine andere Queue)**
- **Secure Queue (nur explizit angegebene User können die Queue nutzen)**

Zusammenfassung: Quellen

- **Database Advanced Queuing User's Guide**

- **https://docs.oracle.com/database/122/ADQUE/aq-introduction.htm#ADQUE0100**


Q&A