PEARSON Education

Markt+Technik

Oracle 10g

Grid Computing, Self-Management, Enterprise Security

FRÖHLICH CZARSKI MAIER

Markt+Technik

KOMPENDIUM

Einführung | Arbeitsbuch | Nachschlagewerk

3 Oracle Streams

Oracle Streams, in der Version 9*i* eingeführt, wurde in Oracle 10g deutlich verbessert. Gleichzeitig fand eine Neuordnung der Produkte statt. Das neue Oracle Streams besteht aus den folgenden Komponenten:

- Oracle Streams
- Oracle Streams Replication
- Oracle Streams Advanced Queuing

Hinter Oracle Streams verbirgt sich die eigentliche Streams-Technologie, die auch in anderen Produkten und Bereichen eingesetzt wird.

Oracle Streams Replication ist das neue Replikationsprodukt in Oracle. Es wurde für Oracle 10g neu entwickelt und basiert auf der Streams-Technologie. Es stellt damit ein Konkurrenzprodukt zu Oracle Advanced Replication dar. Advanced Replication wird in Oracle 10g aus Kompatibilitätsgründen noch mit ausgeliefert. Es findet also in diesem Bereich ein Technologiewechsel statt.

:-)
TIPP

Migrieren Sie sobald wie möglich Anwendungen, die auf Oracle Advanced Replication basieren, nach Oracle Streams Replication. Es ist zu erwarten, dass Advanced Replication in späteren Versionen nicht mehr ausgeliefert wird. Zusätzlich können Sie die Vorteile von Streams Replication nutzen.

Anders verhält es sich für Oracle Streams Advanced Queuing. Dahinter verbirgt sich das Produkt, das Sie als Oracle Advanced Queuing aus der vorangegangenen Version kennen, mit einigen Erweiterungen und Verbesserungen. Advanced Queuing ist im Gegensatz zu Oracle Replication ein wesentlich neueres Produkt.

Oracle Streams kann nunmehr als Technologie für gemeinsame Datenbenutzung angesehen werden und hat sich damit die folgenden Einsatzgebiete erschlossen:

- Event Management
- Replikation von Daten

- Message Queuing
- Data Warehouse ETL
- Data Protection

Mit Oracle Streams ist es also möglich, Nachrichten einzustellen, zu propagieren und auszulesen. Wird Streams für die Replikation eingesetzt, dann werden DML- und DDL-Änderungen von Datenbankobjekten aufgesammelt und den Zieldatenbanken zur Verfügung gestellt.

Darüber hinaus bietet Streams eine Infrastruktur zur Kommunikation an. Im Data-Warehouse-Umfeld stellt Streams eine flexible Technologie für ETL-Prozesse zur Verfügung.

Auch für den Bereich Data Protection ist Oracle Streams eine moderne Alternative. Eine Remote-Datenbank kann mit Änderungsinformationen der Primär-Datenbank gefüttert werden.

:-) TIPP

Logical Standby-Datenbanken werden benutzt, um z.B. eine identische Kopie einer Datenbank zu erzeugen, die im READ ONLY*-Modus läuft und damit für Abfragen zur Verfügung steht. Das entlastet die Primär-Datenbank. Logical Standby-Datenbanken unterliegen jedoch starken Restriktionen für Datentypen, auch gab es in der Vergangenheit immer wieder technische Probleme. Oracle Streams erreicht eine gleichartige Funktionalität und ist damit ein Konkurrenzprodukt zur Logical Standby-Datenbank. Es bietet allerdings eine bessere Flexibilität und einen größeren Funktionsumfang als die Standby-Datenbank.*

Nachdem Sie die neue Streams-Technologie im Detail kennen gelernt haben, werden Sie einige weitere Anwendungsgebiete für Ihre Applikationen entdecken.

i INFO

Es ist auffällig, dass Oracle mit der Version 10g in eine Phase eingetreten ist, in der eine Reihe von langjährig erfolgreichen Technologien durch neue abgelöst werden. Einige der alteingesessenen Technologien genügen den heutigen und zukünftigen Anforderungen nicht mehr. Sie sind damit in der Situation, Lösungen mit gleichartigen Oracle-Produkten realisieren zu können. Entscheiden Sie sich dabei stets für das neue Produkt, um einerseits die Vorteile nutzen zu können und andererseits eine später notwendige Migration zu umgehen.

3.1 Streams-Technologie

Oracle Streams ist eine Technologie zur gemeinsamen Benutzung von Daten. Das gemeinsame Verwenden von Daten in verschiedenen Datenbanken oder in mehreren Anwendungen ist weit verbreitet.

Mit Oracle Streams wird der Datenfluss definiert, durchgeführt und kontrolliert. Änderungen in einer Datenbank werden aufgesammelt, zu den Zielen geleitet und dort konsumiert. Das sind die drei typischen Schritte in Oracle Streams: *Capture*, *Staging* und *Consumption*.

Abbildung 3.1: Die Schritte in Oracle Streams

Die Streams-Architektur

Der Capture-Prozess ist der erste Schritt. Änderungen in einer Datenbank werden in den Online Redo Log-Dateien gespeichert, um die Wiederherstellbarkeit der Datenbank im Fall von Fehlern zu garantieren. Ein Hintergrundprozess liest die Online Redo Log-Dateien und filtert die Änderungen heraus. Diese werden formatiert und in so genannte Events umgewandelt. Ein solches Event wird *Logical Change Record* (LCR) genannt.

Es existieren zwei Arten von LCR, Row LCR und DDL LCR. Diese spiegeln Änderungen wider, die entweder auf DML- oder auf DDL-Basis gemacht wurden. Mit Hilfe von Regeln wird festgelegt, welche Änderungen aufgesammelt werden.

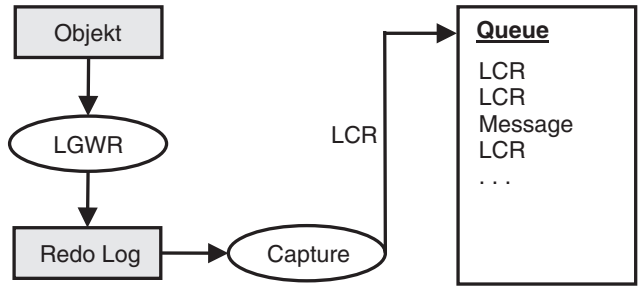
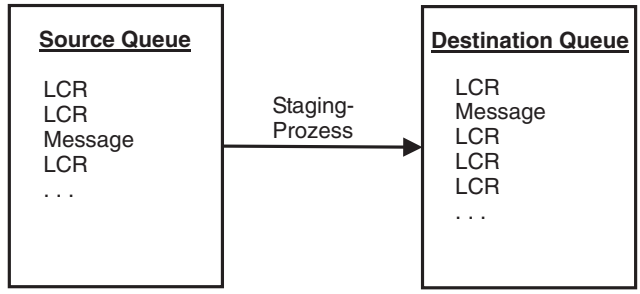


Abbildung 3.2: Der Capture-Prozess für Oracle Streams

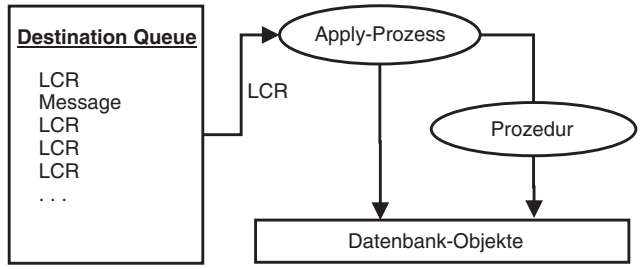
Streams verwendet Queues zum Einstellen von Events. Diese Events werden propagiert, um von Konsumenten verarbeitet zu werden. Zum Propagieren benutzt Streams eine so genannte *Source Queue*, die Queue des Konsumenten heißt *Destination Queue*. Der gesamte Prozess, in dem die Queues verarbeitet und gesteuert werden, wird Staging-Prozess genannt.

Abbildung 3.3: Staging-Prozess in Oracle Streams



Der Konsument der Events von Streams ist ein Messaging Client. Regeln legen fest, welche Events vom Client verarbeitet werden. Das Auslesen der Events erfolgt durch den *Apply-Prozess*. Der Apply-Prozess arbeitet die Änderungen in die Zielobjekte ein. Die Einarbeitung kann direkt oder über eine benutzerdefinierte Prozedur erfolgen.

Abbildung 3.4: Der Apply-Prozess in Oracle Streams



Nach dieser kurzen Architekturbeschreibung erahnen Sie bereits, welche Möglichkeiten sich mit Oracle Streams bieten.

Mit Hilfe von Oracle Transparent Gateway ist es möglich, Datenbanken anderer Hersteller in Oracle Streams einzubinden.

Die Queues in Oracle Streams benutzen einen Queue Buffer im Shared Memory. In früheren Versionen benutzte Streams den Shared Pool, wobei die maximale Ausnutzung auf 10% des Shared Pools begrenzt war.

In Oracle 10g wurde der neue Initialisierungsparameter `STREAMS_POOL_SIZE` eingeführt. Ist sein Wert größer als null, dann benutzt Streams den Streams Pool und nicht den Shared Pool. Der Streams Pool ist Bestandteil der SGA, das heißt, Sie müssen auch den Parameter `SGA_MAX_SIZE` entsprechend erhöhen.

Der Parameter `STREAMS_POOL_SIZE` *ist dynamisch, das heißt, er kann ohne Neustart der Datenbank verändert werden. Wenn Sie den Parameter auf einen Wert größer null gesetzt haben und diesen dynamisch auf null setzen, dann wird der Streams Pool in der SGA entfernt. Allerdings wird kein Spei-*

cher aus dem Shared Pool für Oracle Streams zur Verfügung gestellt, das heißt, Oracle Streams ist damit nicht mehr verfügbar. In diesem Fall muss ein Neustart der Datenbank erfolgen.

Administration von Streams

Dieser Abschnitt beschäftigt sich mit den Aktivitäten zur Vorbereitung einer Datenbank für den Einsatz von Oracle Streams.

Eine Oracle Streams-Umgebung herstellen

Zum Herstellen einer Umgebung für Oracle Streams ist es notwendig, einen Streams-Administrator zu erstellen. Er erhält die erforderlichen Privilegien, um Streams zu verwalten.

Benutzen Sie nicht die User SYS *oder* SYSTEM *als Streams-Administrator. Die Tablespace SYSTEM sollte nicht die Default Tablespace des Administrators sein. Erstellen Sie am besten eine spezielle Tablespace für diesen Zweck oder verwenden Sie* SYSAUX.

Die folgenden Schritte beschreiben, wie ein Streams-Administrator eingerichtet werden kann. Das muss auf jeder Datenbank erfolgen, auf der Sie Streams einsetzen.

Den Streams-Administrator einrichten

1. Erstellen Sie eine Tablespace für den Streams-Administrator.

```
SQL> CREATE TABLESPACE sysstreams
  2  DATAFILE '/u02/oracle/komp10g2/sysstreams01.dbf'
  3  SIZE 30M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
Tablespace wurde angelegt.
```

2. Legen Sie einen neuen Benutzer an, der Streams-Administrator wird.

```
SQL> CREATE USER streams_adm
  2  IDENTIFIED BY streams_adm
  3  DEFAULT TABLESPACE sysstreams;
Benutzer wurde angelegt.
```

3. Geben Sie dem Benutzer die Privilegien CONNECT, RESOURCE und DBA.

```
SQL> GRANT CONNECT, RESOURCE, DBA TO streams_adm;
Benutzerzugriff (Grant) wurde erteilt.
```

4. Der Streams-Administrator benötigt weitere Privilegien, wenn er z.B. benutzerdefinierte Unterprogramme verwendet, die Abfragen über Streams an den Datenbankkatalog stellen. Während in Oracle9*i* die zusätzlichen Privilegien noch einzeln administriert werden mussten,

gibt es in Oracle 10g das Paket DBMS_STREAMS_AUTH. Mit Hilfe der Prozedur GRANT_ADMIN_PRIVILEGE kann ein Benutzer in die Lage versetzt werden, alle Streams-Operationen, wie Capture, Propagation, Queuing, Rules usw. ausführen zu können. Dabei können Sie wahlweise die Privilegien direkt zuweisen oder ein Skript erstellen, das Sie editieren und laufen lassen können.

– Das folgende Beispiel zeigt, wie die Privilegien direkt zugewiesen werden.

```
SQL> BEGIN
  2  DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE
   (grantee=>'streams_adm', grant_privileges=>true);
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

– Alternativ können Sie ein Skript generieren.

```
SQL> CREATE DIRECTORY scripts AS
    '/u01/oracle/admin/komp10g2/scripts';
Verzeichnis wurde erstellt.
SQL> BEGIN
  2  DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE
   (grantee=>'streams_adm', grant_privileges=>false,
   file_name=>'streams_adm_privs.sql', directory_name=>'scripts');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Es gibt eine Reihe von Initialisierungsparametern, die für die Anwendung von Streams gesetzt werden müssen. Passen Sie die Parameter an Ihre Streams-Umgebung an.

Tabelle 3.1: Initialisierungsparameter, die für Oracle Streams benötigt werden

Parameter	Beschreibung
COMPATIBLE	Muss auf 10.1.0.2.0 (bzw. neueste Oracle-Version) gesetzt sein, um die neuen Streams-Features von Oracle 10g benutzen zu können.
PROCESSES	Stellen Sie sicher, dass der Parameter groß genug ist, so dass alle mit Streams verbundenen Prozesse laufen können.
PARALLEL_MAX_SERVERS	In Streams können Capture- und Apply-Prozess parallele Server-Prozesse verwenden. Setzen Sie den Parameter groß genug, um genügend parallele Prozesse zur Verfügung zu haben.

Tabelle 3.1: Initialisierungsparameter, die für Oracle Streams benötigt werden (Forts.)

Parameter	Beschreibung
REMOTE_ARCHIVE_ENABLE	Ermöglicht das Senden und Empfangen von Archived-Redo-Informationen zu und von anderen Datenbanken. Setzen Sie den Parameter auf beiden Seiten auf TRUE, wenn Sie das Feature Downstream Capture verwenden wollen.
SESSIONS	Jeder Hintergrundprozess benötigt eine Sitzung. Setzen Sie den Parameter groß genug, so dass alle Streams-Prozesse eine Verbindung aufbauen können.
SHARED_POOL_SIZE	Falls der Parameter STREAMS_POOL_SIZE auf null gesetzt ist, sollte der Shared Pool ca. 10% größer gemacht werden.
STREAMS_POOL_SIZE	Spezifiziert die Größe des Streams Pools. Er enthält Events und wird für die interne Kommunikation von parallelen Capture- und Apply-Prozessen verwendet. Als Faustregel gilt: 10 Mbyte für jeden Capture-Prozess, 1 Mbyte für jeden Apply-Prozess, 10 Mbyte für jedes Queue Staging Event.
TIMED_STATISTICS	Setzen Sie den Parameter auf TRUE, wenn Sie Performance-Statistiken über Streams sammeln wollen.
UNDO_RETENTION	Der Parameter sollte groß genug sein, um den Fehler »snapshot too old« zu vermeiden. Für den Einsatz von Oracle Streams sollte er mindestens auf 3600 stehen.
JOB_QUEUE_PROCESSES	Sollte für Streams mindestens auf »2« gesetzt werden. Als Faustregel gilt: max. Anzahl von Jobs, die parallel laufen, plus zwei.
LOG_ARCHIVE_DEST_n	Muss gesetzt werden, wenn Sie das Feature Downstream Capture verwenden.
LOG_ARCHIVE_DEST_STATE_n	Muss gesetzt werden, wenn Sie das Feature Downstream Capture verwenden.
OPEN_LINKS	Setzen Sie den Parameter für Streams mindestens auf vier.
GLOBAL_NAMES	Muss für alle Datenbanken, die in Streams eingebunden sind, auf TRUE gesetzt werden.

Ein Capture-Prozess benutzt immer Online Redo Log-Informationen und greift nur auf Archived Redo Log-Dateien zurück, wenn das nicht möglich ist. Ein Downstream Capture-Prozess liest immer aus den Archived Redo Log-Dateien der Quelldatenbank. Deshalb muss die Datenbank im ARCHIVELOG*-Modus laufen und es müssen immer die notwendigen Archived Redo Log-Dateien zur Verfügung stehen.*

Stellen Sie außerdem sicher, das die in Oracle Streams eingebundenen Datenbanken gegenseitig über Oracle Net zu erreichen und mit Datenbank-Links untereinander verbunden sind. Der Datenbank-Link wird für administrative Zwecke eingesetzt.

Den Capture-Prozess verwalten

Es existieren zwei Optionen, einen Capture-Prozess zu erzeugen:

- ein Capture-Prozess, der Änderungen einer lokalen Datenbank aufsammelt
- ein Capture-Prozess, der Änderungen aus der Ferne von einer Downstream Database aufnimmt

Wenn der Capture-Prozess auf einer Downstream Database läuft, dann werden die Redo Log-Dateien von der Quell- zur Downstream-Datenbank kopiert und dort vom Capture-Prozess aufgenommen. Das Erzeugen eines Capture-Prozesses erfolgt mit den Prozeduren des Paketes `DBMS_STREAMS_ADM`.

Bevor Sie jedoch einen Capture-Prozess erzeugen können, müssen Sie eine Queue Table für Streams erstellen. Verwenden Sie dafür die Prozedur `SET_UP_QUEUE` wie im folgenden Beispiel.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.SET_UP_QUEUE( queue_table=>'streams_queue_table',
queue_name=>'streams_queue');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Die Queue Table wurde als Buffered Queue Table erstellt. Buffered Queues besitzen signifikante Performance-Vorteile gegenüber herkömmlichen Queues. Das neue View `V$BUFFERED_QUEUES` beschreibt alle Buffered Queues. Die Spalten haben folgende Bedeutung:

- `NUM_MSGS`: Anzahl der Nachrichten in der Queue
- `SPILL_MSGS`: Anzahl der Nachrichten in der Queue, die auf Platte geschrieben wurden
- `CNUM_MSGS`: Kumulative Anzahl der Nachrichten in der Queue
- `CSPILL_MSGS`: Kumulative Anzahl der Nachrichten in der Queue, die auf Platte geschrieben wurden

```
SQL> SELECT * FROM v$buffered_queues;
  QUEUE_ID QUEUE_SCHEMA QUEUE_NAME    STARTUP_   NUM_MSGS SPILL_MSGS
CNUM_MSGS CSPILL_MSGS
```

```
---------- ------------ ------------- -------- ---------- ----------
---------- -----------
     11430 STREAMS_ADM  STREAMS_QUEUE 09.05.04          0          0
         0           0
```

Das folgende Beispiel erstellt einen Local-Capture-Prozess.

```
SQL> BEGIN
  2  DBMS_CAPTURE_ADM.CREATE_CAPTURE( queue_name=>'streams_queue',
capture_name=>'local_capture', start_scn=>null, source_database=>null,
use_database_link=>false, first_scn=>null);
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Der Capture-Prozess kann mit Hilfe des Paketes `DBMS_STREAMS_ADM` gestartet und gestoppt werden.

```
SQL> BEGIN
  2  DBMS_CAPTURE_ADM.START_CAPTURE( capture_name=>'local_capture');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
SQL> BEGIN
  2  DBMS_CAPTURE_ADM.STOP_CAPTURE( capture_name=>'local_capture');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Die Verwaltung von Oracle Streams ist noch nicht im neuen Enterprise Manager integriert. Benutzen Sie stattdessen die Enterprise Manager-Java-Konsole.

:-) TIPP

Staging und Propagation

Das Staging erfolgt über die Queue Tables. Wie bereits erwähnt, überträgt der Staging-Prozess die Nachrichten aus der Source-Queue in die Destination-Queue.

Der Prozess der Übertragung von der Source-Queue in die Destination-Queue wird auch als *Propagation* bezeichnet. Mit Hilfe des Paketes `DBMS_STREAMS_ADM` können Sie eine Propagation erstellen. Dabei werden so genannte Rules definiert. Diese legen fest, wie die Propagation erfolgen soll.

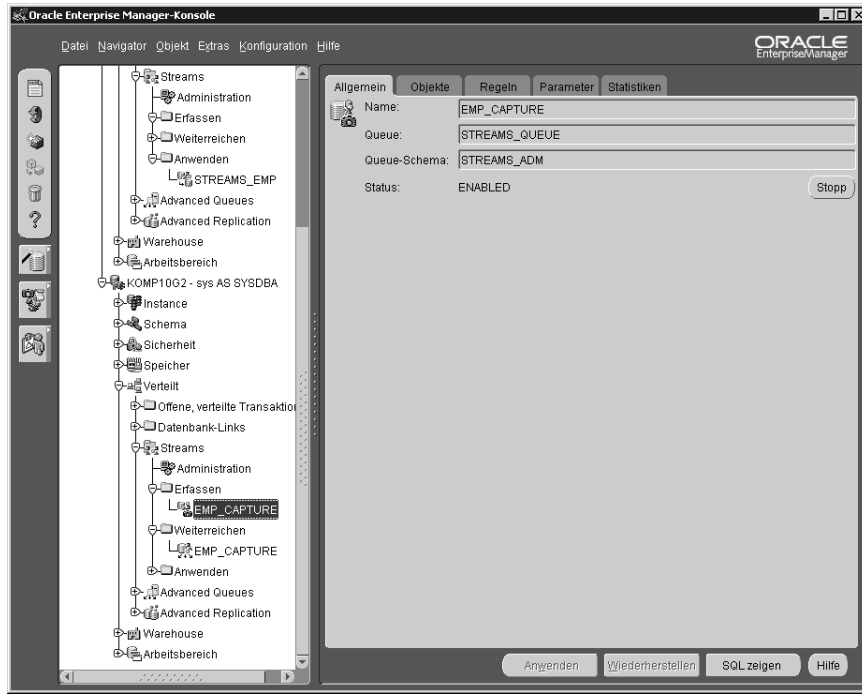


Abbildung 3.5: Verwaltung von Oracle Streams im Enterprise Manager

Das folgende Beispiel zeigt, wie eine Propagation erstellt werden kann.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES( table_name=>'komp10g.emp',
 streams_name=>'emp_stream', source_queue_name=>'streams_adm.streams_queue',
destination_queue_name=>'streams_queue', include_dml=>true,
include_ddl=>true, include_tagged_lcr=>false, source_database=>'komp10g2',
inclusion_rule=>true);
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Der Apply-Prozess

Der Apply-Prozess sorgt für das Einarbeiten der Events, die in der Destination-Queue als LCR vorliegen, in die Zieldatenbank. Das Starten des Apply-Prozesses erfolgt mit den Prozeduren der Pakete `DBMS_STREAMS_ADM` und `DBMS_APPLY_ADM`.

Auch für den Apply-Prozess können Rules definiert werden. Das folgende Beispiel zeigt, wie ein Apply-Prozess gestartet werden kann.

```
SQL> BEGIN
  2  DBMS_APPLY_ADM.CREATE_APPLY( queue_name=>'streams_queue', apply_name=>'s
treams_apply', apply_user=>'streams_adm', apply_captured=>true,source_databas
```

```
e=>'komp10g2');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Ein komplettes Beispiel für Local Streams Capture

Eine Oracle Streams-Architektur bietet den Vorteil, dass sie einerseits sehr flexibel ist. Der Nachteil der vielen Optionen in den einzelnen Teilbereichen Capture, Staging und Applying ist, dass Erstellung und Konfiguration recht aufwendig sind. Die Oracle-Terminologie ist zudem teilweise verwirrend.

Das soll uns jedoch nicht davon abhalten, diese sehr nützliche und zukunftsweisende Technologie zu verwenden. Dieser Abschnitt beschreibt ein komplettes Beispiel, das Sie nachempfinden und auf Ihre Belange anpassen können.

Auf der Source-Datenbank sollen alle Ereignisse, die Veränderungen der Tabelle `komp10g.emp` hervorrufen, aufgesammelt werden. Diese werden über Staging- und Apply-Prozess in die Destination-Datenbank eingearbeitet. In Abbildung 3.6 finden Sie eine bildliche Darstellung der gewünschten Architektur.

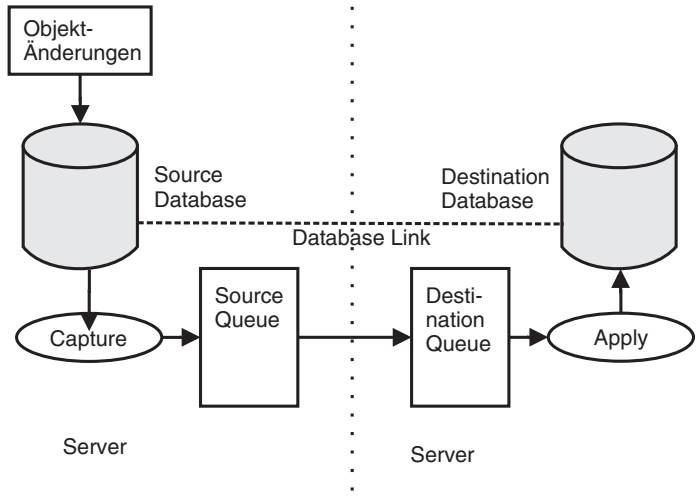


Abbildung 3.6: Beispiel für Local Streams Capture

Die Konfiguration besteht aus zwei Teilen. Im ersten Teil erfolgt die Konfiguration der Destination Database, im zweiten die der Source Database. Dabei spielt es keine Rolle, ob die Datenbanken auf einem oder verschiedenen Servern laufen. Im vorliegenden Beispiel laufen die Datenbanken auf getrennten Servern.

Konfiguration von Oracle Streams auf der Destination Database

1. Öffnen Sie eine SQL*Plus-Sitzung als Streams-Administrator auf der Destination Database.

```
SQL> CONNECT streams_adm/streams_adm@komp10g
Connect durchgeführt.
```

2. Erstellen Sie eine Destination-Queue für Streams.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.SET_UP_QUEUE( queue_table=>'streams_queue_table',
    queue_name=>'streams_queue');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

3. Definieren Sie die Rules für den Apply-Prozess auf der Destination Database. Es sollen alle DML-Änderungen für die Tabelle `komp10g.emp` eingearbeitet werden. Änderungen in der DDL sollen nicht berücksichtigt werden.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.ADD_TABLE_RULES( table_name=>'komp10g.emp',
    streams_type=>'apply', streams_name=>'streams_emp',
    queue_name=>'streams_adm.streams_queue',
    include_dml=>true, include_ddl=>false, source_database=>'komp10g2.wor
    ld');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Führen Sie die folgenden Schritte auf der Source Database aus.

Konfiguration von Oracle Streams auf der Source Database

1. Öffnen Sie eine SQL*Plus-Sitzung als Streams-Administrator.

```
SQL> CONNECT streams_adm/streams_adm@komp10g2
Connect durchgeführt.
```

2. Schalten Sie Supplemental Logging ein, um die Datenintegrität zu gewährleisten.

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY,
   UNIQUE INDEX) COLUMNS;
Datenbank wurde geändert.
```

3. Führen Sie einen Log File Switch durch.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
System wurde geändert.
```

4. Erstellen Sie einen Datenbank-Link zur Destination Database. Verwenden Sie dafür den Benutzer `streams_adm`.

```
SQL> CREATE DATABASE LINK komp10g.world
  2  CONNECT TO streams_adm
  3  IDENTIFIED BY streams_adm
  4  USING
  5  '(DESCRIPTION=ADDRESS_LIST = (ADDRESS = (PROTOCOL=TCP)
    (HOST=mch1.pitcorp.com) (PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=komp10g.world)))';
Datenbank-Link wurde angelegt.
```

5. Legen Sie eine Streams-Queue auf der Source-Datenbank an und definieren Sie die Rules für den Capture-Prozess. Es sollen DML-Änderungen in der Tabelle `komp10g.emp` erfasst werden.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.SET_UP_QUEUE( queue_table=>'streams_queue_tables',
     queue_table_name=>'streams_queue');
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
SQL> BEGIN
  2  DBMS_STREAMS_ADM.ADD_TABLE_RULES( table_name=>'komp10g.emp',
    streams_type=>'capture', streams_name=>'emp_capture',
    queue_name=>'streams_adm.streams_queue', include_dml=>true,
    include_ddl=>false, source_database=>'komp10g2.world');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

6. Legen Sie die Propagation Rules für Oracle Streams fest.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES(
    table_name=>'komp10g.emp', streams_name=>'emp_capture',
    source_queue_name=>'streams_adm.streams_queue',
    destination_queue_name=> 'streams_adm.streams_queue@komp10g.world',
    include_dml=>true, include_ddl=>false,
    source_database=>'komp10g2.world');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

7. Führen Sie zur Synchronisation einen Export und einen Import der Tabelle `emp` durch. Dies kann auf dem Server mit der Source Database erfolgen.

```
exp userid=streams_adm/
   streams_adm@komp10g2 tables=komp10g.emp file=tables.dmp grants=y
   rows=y log=tables.log object_consistent=y indexes=y
Export: Release 10.1.0.2.0 - Production on Mo Mai 10 11:59:18 2004
Copyright (c) 1982, 2004, Oracle.  All rights reserved.
```

```
Verbunden mit: Oracle Database 10g Enterprise Edition Release 10.1.0.2.0
    - Production
With the Partitioning, OLAP and Data Mining options
Exportieren in WE8ISO8859P1-Zeichensatz und AL16UTF16-NCHAR-
   Zeichensatz durchgeführt
Angegebene Tabellen werden gleich exportiert über 'Conventional Path'
Derzeitiger Benutzer ist auf KOMP10G gewechselt
. . Export der Tabelle                          EMP         10 Zeilen
   exportiert
Export erfolgreich und ohne Warnungen beendet.
imp streams_adm/
   streams_adm@komp10g full=y constraints=y file=tables.dmp ignore=y
   grants=y rows=y commit=y log=imptables.log streams_configuration=n
   streams_instantiation=y
Import: Release 10.1.0.2.0 - Production on Mo Mai 10 12:03:14 2004
Copyright (c) 1982, 2004, Oracle.  All rights reserved.
Verbunden mit: Oracle Database 10g Enterprise Edition Release 10.1.0.2.0
    - Production
With the Partitioning, Oracle Label Security, OLAP and Data Mining
   options
      <
Export-Datei wurde von EXPORT:V10.01.00 über konventionellen Pfad
   erstellt
Importvorgang mit Zeichensatz WE8ISO8859P1 und
   Zeichensatz AL16UTF16 NCHAR durchgeführt
Import-Server verwendet Zeichensatz WE8ISO8859P15
   (mögliche Zeichensatzkonvertierung)
. Import STREAMS_ADM's Objekte in STREAMS_ADM
. Import KOMP10G's Objekte in KOMP10G
. . Import der Tabelle                        "EMP"         10 Zeilen
   importiert
Der Import-Vorgang endete erfolgreich und ohne Warnungen.
```

8. Auf der Tabelle in der Destination Database muss die *Instantiation SCN* gesetzt werden. Alle LCRs der Source Database, deren SCN höher als die der Instantiation SCN sind, werden auf die Tabelle in der Destination Database angewandt. Die Instantiation SCN wird zwar in einigen Fällen, wie bei `exp` oder `imp` automatisch gesetzt, manchmal ist jedoch auch das manuelle Setzen erforderlich. Selektieren Sie die aktuelle SCN der Source Database, in dem Sie die Spalte `CURRENT_SCN` in dem View `V$DATABASE` und wenden Sie diese in der Destination Database an.

```
SQL> BEGIN
  2  DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN(
source_object_name=>'komp10g.emp',
   source_database_name=>'komp10g2.world', instantiation_scn=>6554323);
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Damit sind die Konfigurationsschritte auf Source Database und Destination Database abgeschlossen. Jetzt können der Apply-Prozess auf der Destination Database und der Capture-Prozess auf der Source Database gestartet werden.

Listing 3.1: Den Apply-Prozess auf der Destination Database starten

```
SQL> DECLARE
  2  v_started NUMBER;
  3  BEGIN
  4  SELECT DECODE(status, 'ENABLED', 1, 0)
  5  INTO v_started
  6  FROM DBA_APPLY WHERE APPLY_NAME='STREAMS_EMP';
  7  IF (v_started = 0) THEN
  8    DBMS_APPLY_ADM.START_APPLY( apply_name=>'streams_emp');
  9  END IF;
 10  END;
 11  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Listing 3.2: Den Capture-Prozess auf der Source Database starten

```
SQL> DECLARE
  2  v_started NUMBER;
  3  BEGIN
  4  SELECT DECODE(status, 'ENABLED', 1, 0)
  5  INTO v_started
  6  FROM dba_capture
  7  WHERE capture_name='EMP_CAPTURE';
  8  IF (v_started = 0) THEN
  9    DBMS_CAPTURE_ADM.START_CAPTURE( capture_name=>'emp_capture');
 10  END IF;
 11  END;
 12  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Wenn Capture- und Apply-Prozess gestartet sind, sehen Sie diese als Hintergrundprozesse der Oracle-Datenbank. Die Capture-Prozesse besitzen die Namen `cnnn`*, die Apply-Prozesse die Namen* `annn`*. Die Anzahl der Prozesse hängt vom Grad der Parallelisierung ab.*

```
$ ps -ef|grep ora_c0
oracle    3996     1  0 12:20 ?        00:00:03 ora_c001_komp10g2
$ ps -ef|grep ora_a0
oracle    1686     1  0 12:15 ?        00:00:00 ora_a001_komp10g
```

Damit ist Oracle Streams in der gewählten Architektur vollständig konfiguriert und aktiviert. Überprüfen Sie nun, ob DML-Änderungen in der Source Database in der Destination Database ankommen. Fügen Sie dazu einen Satz in die Tabelle `emp` ein.

```
SQL> CONNECT streams_adm/streams_adm@komp10g2
Connect durchgeführt.
SQL> INSERT INTO komp10g.emp VALUES
  2  (4711, 'STREAMS', 'MANAGER', 7839, sysdate, 1200, 100, 10);
1 Zeile wurde erstellt.
SQL> COMMIT;
Transaktion mit COMMIT abgeschlossen.
SQL> CONNECT streams_adm/streams_adm@komp10g
Connect durchgeführt.
SQL> SELECT * FROM komp10g.emp
  2  WHERE empno = 4711;
    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
 DEPTNO
---------- ---------- --------- ---------- -------- ---------- ----------
----------
      4711 STREAMS    MANAGER         7839 10.05.04       1200        100
        10
```

Sie werden feststellen, dass die Aktualisierung der Destination Database zwar asynchron, aber recht zeitnah (in wenigen Sekunden) erfolgt. Nach der `COMMIT`-Anweisung führt Oracle Streams sofort Capture, Staging und Consumption durch. Die Geschwindigkeit hängt im Wesentlichen von der Geschwindigkeit der Queues sowie der Übertragungsgeschwindigkeit über das Netzwerk ab.

Kommt es zu Fehlern oder Problemen, dann müssen Sie Troubleshooting betreiben. Hier empfiehlt es sich, schrittweise vorzugehen, in der Reihenfolge, in der die Oracle Streams-Prozesse eingebunden sind. Wichtige Hinweise beinhalten die Views `DBA_CAPTURE` und `DBA_APPLY`, aber auch die Source- und Destination-Queues.

```
SQL> SELECT capture_name, status, error_number
  2  FROM dba_capture;
CAPTURE_NAME                   STATUS   ERROR_NUMBER
------------------------------ -------- ------------
EMP_CAPTURE                    ENABLED
SQL> SELECT apply_name, status, error_number
  2  FROM dba_apply;
APPLY_NAME                     STATUS   ERROR_NUMBER
------------------------------ -------- ------------
STREAMS_EMP                    ENABLED
```

Auch der Enterprise Manager bietet wertvolle Informationen und Statistiken. Beachten Sie, dass Oracle Streams noch nicht in den neuen webbasierenden Enterprise Manager integriert ist. Sie müssen also die Java-Konsole verwenden. In Abbildung 3.7 sehen Sie eine transaktionsbezogene Statistik des Apply-Prozesses. Die Features für Oracle Streams finden Sie unter den Verzweigungen VERTEILT/STREAMS.

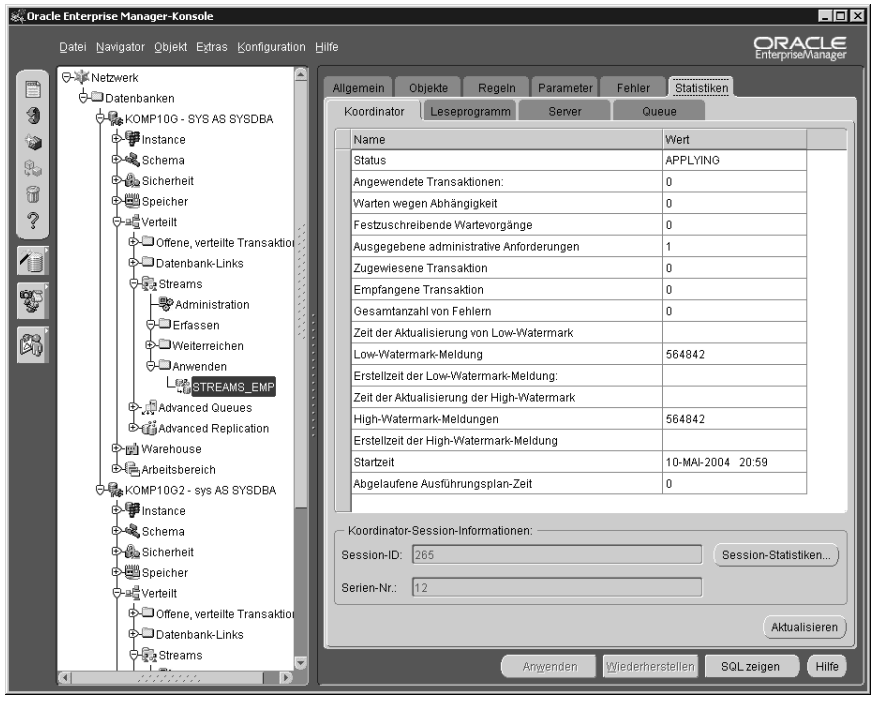


Abbildung 3.7: Statistik des Apply-Prozesses von Oracle Streams im Enterprise Manager

Recht nützlich ist auch die Topologie-Übersicht, die Sie unter STREAMS/ADMINISTRATION finden.

Downstream Capture

Downstream Capture ist ein neues Feature in Oracle 10g. Normalerweise läuft in Oracle Streams der Capture-Prozess auf der Source Database. Mit Downstream Capture ist es möglich, das Capturing auf einer anderen Datenbank durchzuführen.

Die Datenbank, auf der der Capture-Prozess läuft, wird Downstream Database genannt. Die Destination Database kann sich auf einem beliebigen Server befinden, nicht zwangsläufig auf dem Downstream Server. Abbildung 3.9 illustriert die Architektur von Downstream Capture.

Downstream Capture eliminiert mögliche Performance-Engpässe in der Source Database.

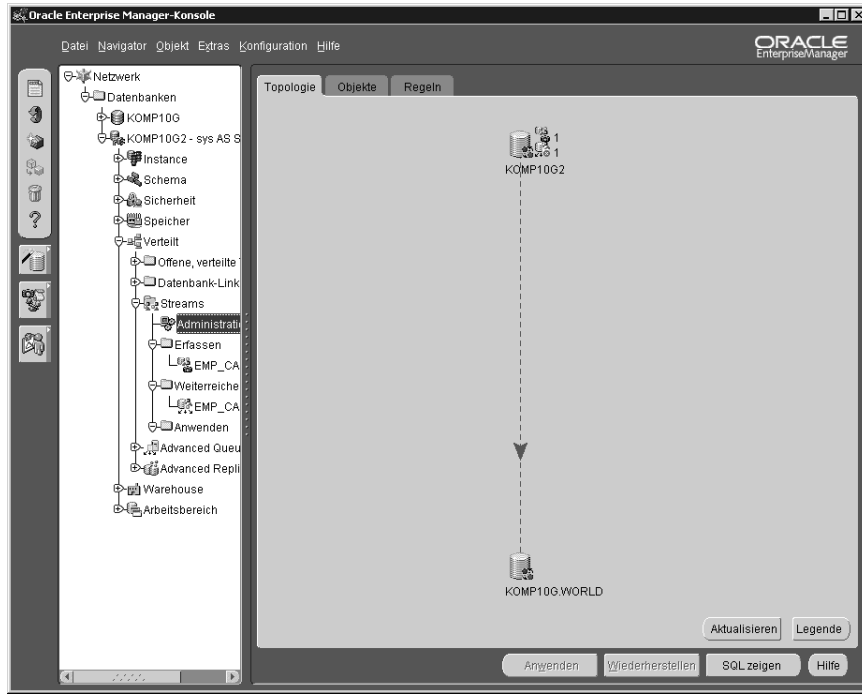


Abbildung 3.8: Topologie-Übersicht von Oracle Streams im Enterprise Manager

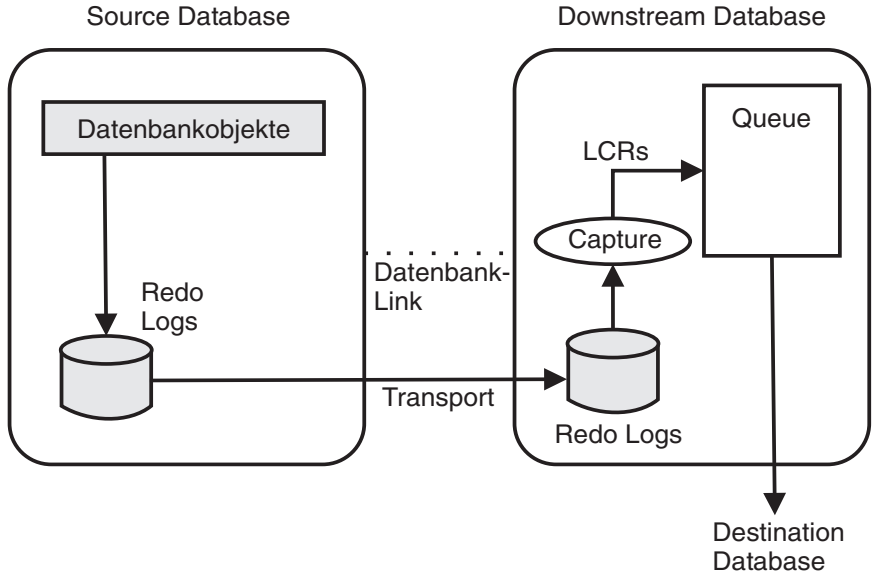


Abbildung 3.9: Die Architektur von Oracle Streams Downstream Capture

Da der Capture-Prozess nicht in der Source Database läuft, entlastet dies die Source Database. Die Archived Redo Log-Dateien werden von der Source Database zur Downstream Database transportiert. Für den Transport können Sie entweder den Log Transport Service der Datenbank, das Paket

DBMS_FILE_TRANSFER oder eine beliebige andere Methode wie z.B. FTP oder SCP verwenden.

Danach läuft der Prozess wieder wie gewohnt ab. Der Capture-Prozess sammelt die relevanten Änderungen auf, erzeugt LCRs und schreibt diese in eine Queue. Danach kann der Apply-Prozess erfolgen.

Unterscheiden Sie unbedingt zwischen Downstream Database und Destination Database. Die Downstream Database ist mit dem Capture-Prozess verbunden. Sie ist zwar in der Praxis häufig mit der Destination Database identisch, es ist aber nicht notwendig. Die Destination Database kann eine andere Datenbank sein und kann sich auf einem anderen Server befinden. Mit der Streams-Technologie wird dann die Destination Queue gefüllt und der Apply-Prozess fügt die Änderungen in die Destination Database ein.

Downstream Capture verfügt neben Entlastung der Source Database über weitere Vorteile:

- Eine Datenbank kann mehrere Capture-Prozesse von mehreren Source Databases ersetzen. Damit ist es möglich, einen einheitlichen Capture-Prozess für alle Source Databases zu definieren.
- Da sich die Redo Log-Dateien zusätzlich auf der Downstream Database befinden, besitzen Sie eine zusätzliche Sicherheitskopie Ihrer Redo Log-Dateien. Dieser zusätzliche Sicherheitsfaktor kann mit einer Standby-Datenbank verglichen werden.

Wie immer, gibt es auch Einschränkungen beim Einsatz einer Downstream Database:

- Sowohl Source Database als auch Downstream Database müssen die Version 10g besitzen.
- Beide Datenbanken müssen mit demselben Betriebssystem laufen, die Version kann jedoch verschieden sein.
- Die Hardware- und Betriebssystem-Architektur muss identisch sein. Sie können nicht 32-bit und 64-bit mischen.
- Die Source Database und die Downstream Database müssen jeweils über eigene Kontrolldateien verfügen.
- Nachdem der Capture-Prozess auf der Downstream Database erzeugt wurde, können der Datenbankname oder die Datenbank-ID der Source Database nicht verändert werden.

Das folgende Beispiel beschreibt, wie eine Downstream Database erstellt werden kann.

Erstellen einer Downstream Database

1. Konfigurieren Sie Oracle Net, so dass die Source Database mit der Downstream Database kommunizieren kann. Richten Sie den Transport Service für das Übertragen der Redo Log-Dateien ein. Sowohl Online- als auch Archived Redo Log-Dateien können für die Übertragung verwendet werden. In diesem Szenario werden die Archived Redo Log-Dateien benutzt.

```
SQL> CONNECT sys/manager@komp10g as sysdba
Connect durchgeführt.
SQL> ALTER SYSTEM SET
  2  log_archive_dest_2='SERVICE=komp10g.world ARCH MANDATORY NOREGISTER
     REOPEN=120
    TEMPLATE=/var/oracle/oradata/komp10g/archive2/
    downstream_%t_%s_%r.dbf';
System wurde geändert.
SQL> ALTER SYSTEM SET
  2  log_archive_dest_state_2=ENABLE;
System wurde geändert.
```

2. Schalten Sie das Supplemental Logging in der Source Database ein.

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA
  2  (PRIMARY KEY, UNIQUE INDEX) COLUMNS;
Datenbank wurde geändert.
```

3. Erstellen Sie einen Datenbank-Link von der Downstream Database zur Source Database.

```
SQL> CONNECT streams_adm/streams_adm@komp10g
Connect durchgeführt.
SQL> CREATE DATABASE LINK komp10g2.world
  2  CONNECT TO streams_adm
  3  IDENTIFIED BY streams_adm
  4  USING 'komp10g2';
Datenbank-Link wurde angelegt.
```

4. Für den Capture-Prozess wird eine Queue auf der Downstream Database benötigt. Erstellen Sie diese mit der Prozedur `SET_UP_QUEUE`.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.SET_UP_QUEUE( queue_table=>'streams_queue_table',
    queue_name=>'streams_queue');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

5. Starten Sie den Capture-Prozess.

```
SQL> BEGIN
  2   DBMS_CAPTURE_ADM.CREATE_CAPTURE( queue_name=>'streams_queue',
    capture_name=>'down_capture', source_database=>'komp10g2.world',
    start_scn=>NULL, use_database_link=>TRUE,
    logfile_assignment=>'implicit');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

6. Definieren Sie die Rules für den Capture-Prozess.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.ADD_TABLE_RULES( table_name=>'komp10g.emp',
    streams_type=>'capture', streams_name=>'down_capture',
    queue_name=>'streams_queue', include_dml=>TRUE, include_ddl=>FALSE,
    source_database=>'komp10g2.world', inclusion_rule=>TRUE);
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

7. Oracle verwendet Informationen aus dem Datenbankkatalog, um einen Log Miner-Katalog in der Downstream Database zu erzeugen. Führen Sie die Prozedur `BUILD` in der Source Database aus, um die Katalog-Informationen in die Redo Log-Dateien zu übertragen.

```
SQL> BEGIN
  2  DBMS_CAPTURE_ADM.BUILD;
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

8. An dieser Stelle können Sie den Propagation- oder den Apply-Prozess definieren und starten. Selbstverständlich können Sie auch beide Prozesse konfigurieren. Im vorliegenden Beispiel wird ein Apply-Prozess erstellt und gestartet.

```
SQL> BEGIN
  2  DBMS_APPLY_ADM.CREATE_APPLY( queue_name=>'streams_queue',
    apply_name=>'down_apply', apply_captured=>TRUE);
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
SQL> BEGIN
  2  DBMS_APPLY_ADM.START_APPLY( apply_name=>'down_apply');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

9. Starten Sie zum Schluss noch den Capture-Prozess.

```
SQL> BEGIN
  2  DBMS_CAPTURE_ADM.START_CAPTURE( capture_name=>'down_capture');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Damit ist die Konfiguration der Downstream-Architektur abgeschlossen und aktiviert. Überprüfen Sie, ob die aufgesetzte Konfiguration funktioniert.

```
SQL> CONNECT streams_adm/streams_adm@komp10g2
Connect durchgeführt.
SQL> INSERT INTO komp10g.emp VALUES
  2  (4712, 'DOWNSTREAM', 'MANAGER', 7839, sysdate, 1500, 50, 10);
1 Zeile wurde erstellt.
SQL> COMMIT;
Transaktion mit COMMIT abgeschlossen.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System wurde geändert.
SQL> CONNECT streams_adm/streams_adm@komp10g
Connect durchgeführt.
SQL> SELECT * FROM komp10g.emp
  2  WHERE empno = 4712;
     EMPNO ENAME      JOB              MGR HIREDATE        SAL       COMM
 DEPTNO
---------- ---------- --------- ---------- -------- ---------- ----------
----------
      4712 DOWNSTREAM MANAGER         7839 11.05.04       1500         50
        10
```

:-) TIPP

Der Logfile Switch im Test ist erforderlich, da im vorliegenden Beispiel die Archived Redo Log-Dateien für den Transport benutzt werden. Sie können auf die Online Redo Log-Dateien verwenden. Dann würde das Einarbeiten in die Destination Database zeitnaher erfolgen.

Wie Sie sicher erkannt haben, ist die Wirkungsweise von Downstream Capture einer Standby-Datenbank sehr ähnlich. Oracle Streams kann durchaus als Konkurrenzprodukt zu Data Guard angesehen werden.

INFO

Ein zusätzlicher Vorteil von Oracle Streams gegenüber Data Guard besteht darin, dass die Destination Database für Anwender geöffnet ist, sich also im OPEN*-Modus befindet. Das ist im Data Guard nur eingeschränkt möglich. Wenn Sie eine physische Standby-Datenbank öffnen, muss der Apply-Prozess gestoppt werden. Eine logische Standby-Datenbank kann zwar parallel im* READ ONLY*-Modus laufen, jedoch gibt es hier relativ viele Einschränkungen, z.B. bezüglich Datentypen. Eine Downstream Database kann bei fort-*

laufendem Apply-Prozess permanent geöffnet sein. Beachten Sie jedoch, dass Streams nicht alle Datentypen unterstützt. So werden z.B. selbst definierte Objekttypen, BFILE- oder auch XMLTYPE-Typen nicht unterstützt.

Die Überwachung von Downstream Capture erfolgt mit den Mitteln, die Oracle Streams zur Verfügung stellt. Das sind Views im Datenbankkatalog sowie die Java-Konsole des Enterprise Manager.

Mit Hilfe des Views `DBA_CAPTURE` können Sie herausfinden, welche Downstream Capture-Prozesse konfiguriert sind.

Listing 3.3: Abfrage der Source Databases mit Downstream Capture

```
SQL> SELECT capture_name, queue_name,
  2  source_database, status
  3  FROM dba_capture
  4  WHERE source_database !=
  5  (SELECT * FROM global_name);
CAPTURE_NAME QUEUE_NAME    SOURCE_DATABASE STATUS
------------ ------------- --------------- --------
DOWN_CAPTURE STREAMS_QUEUE KOMP10G2.WORLD  ENABLED
```

Im Enterprise Manager können Sie sich die Statistiken des Capture-Prozesses auf der Downstream Database anschauen. In Abbildung 3.10 ist der Status `WAITING FOR REDO`. Das bedeutet, der Capture-Prozess ist aktiviert und wartet auf die nächsten Redo Log-Informationen. Bei Problemen und Fehlern gibt Ihnen der Status wichtige Hinweise.

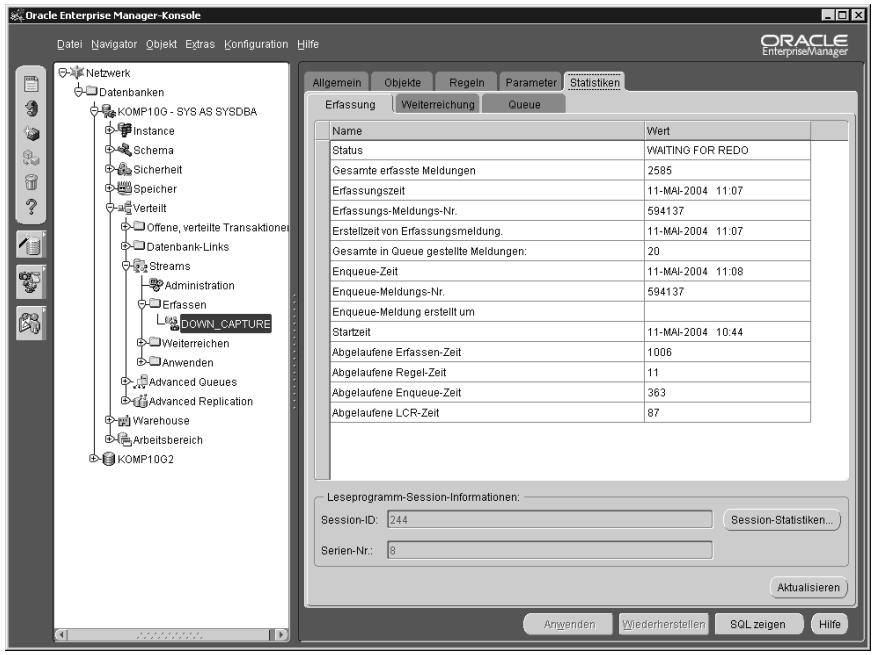


Abbildung 3.10: Statistiken des Apply-Prozesses einer Downstream Database

3.2 Oracle Streams Replication

Oracle Streams Replication ist ein neues Produkt in Oracle 10*g*. Es ist eine Replikationslösung, die auf Oracle Streams-Technologie aufsetzt. Parallel dazu wird Oracle Advanced Replication aus Kompatibilitätsgründen noch mit ausgeliefert. Es ist jedoch zu empfehlen, nach Streams Replication zu migrieren, da Advanced Replication voraussichtlich in späteren Versionen nicht mehr ausgeliefert wird. Oracle bietet bereits Migrationsunterstützung an.

Da Streams Replication auf Streams-Technologie basiert, besteht die Architektur auch hier aus den drei Schritten Capture-Prozess, Propagation und Apply-Prozess (siehe Abbildung 3.1). Diese Prozesse laufen asynchron, die Übermittlung der Änderungsinformationen (LCRs) erfolgt mit Hilfe von Streams Advanced Queuing. In jedem der drei Prozesse können Regeln definiert werden.

Im Vergleich mit Advanced Replication weist Streams Replication die folgenden Vorteile und neuen Features aus:

- Unterstützung von Real Application Clusters
- Erstellen einer Datenbank mit RMAN oder Transportable Tablespaces
- Zusätzliche Attribute in den Logical Change Records (LCR)
- Neue Funktionen und Prozeduren
- Unterstützung für mehr Datentypen und Index Organized Tables

Konfiguration einer Umgebung

Die Konfiguration einer Streams Replication-Umgebung ist abhängig von der Architektur, die Sie einsetzen wollen. Oracle Streams Replication unterscheidet zwischen Single-Source- und Multiple Source-Umgebung. Es ist auch möglich, mehrere Destination Databases zu definieren. In diesem Abschnitt finden Sie eine Anleitung zum Erstellen einer Single-Source-Umgebung.

Enttäuschend ist die Unterstützung, die Oracle für die Verwaltung der Streams-Replication-Architekturen liefert. Es existiert kein Programm mit grafischer Oberfläche (abgesehen von den Streams-Verwaltungsfenstern im Enterprise Manager). Beim genauen Hinsehen muss man feststellen, dass Oracle Streams Replication eigentlich kein Produkt ist. Es basiert auf Streams und man muss sich die Replication-Architekturen selbst mit der Streams-Funktionalität zusammenbasteln. Das trifft auch für das Monitoring zu oder die Darstellungsmöglichkeiten von Topologien. An dieser Stelle besteht noch erheblicher Nachholbedarf.

Gut hingegen sind die Möglichkeiten für die so genannte *Instantiation*. Unter Instantiation versteht man das Einrichten von Ziel-Objekten oder einer ganzen Zieldatenbank. Wenn Sie mit einer Replikationsumgebung starten wollen, dann fehlen Ihnen in der Regel die Objekte und die Datenbanken, in die Sie replizieren wollen. Diese Objekte werden in der Terminologie von Advanced Replication als Materialized Views bezeichnet.

Für die Instantiation können folgende Werkzeuge eingesetzt werden, abhängig von den Objekten, die Sie erzeugen wollen:

- RMAN
- Transportable Tablespaces
- Data Pump

Wenn Sie eine neue Datenbank erstellen wollen, in die Sie hineinreplizieren, dann empfiehlt sich die Verwendung von RMAN. Die folgenden Schritte beschreiben, wie Sie RMAN für Instantiation benutzen können.

Instantiation von Streams Replication mit RMAN

1. Verwenden Sie den Recovery Manager zum Erstellen einer Komplettsicherung.

```
$ rman target / catalog rman/rman@rman10
Recovery Manager: Version 10.1.0.2.0 - Production
Copyright (c) 1995, 2004, Oracle.  All rights reserved.
Mit Ziel-Datenbank verbunden: KOMP10G2 (DBID=3497543750)
Verbindung mit Datenbank des Recovery-Katalogs
RMAN> BACKUP DATABASE ARCHIVELOG ALL;
Starten backup um 12.05.04
. . .
```

2. Konfigurieren Sie einen Capture-Prozess auf der Source Database.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.ADD_GLOBAL_RULES( streams_type=>'capture',
    streams_name=>'replication',
    queue_name=>'streams_adm.streams_queue', include_dml=>true,
    include_ddl=>true, inclusion_rule=>true);
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

3. Bereiten Sie die Source Database für die Instantiation vor. Mit dem Ausführen der Prozedur `PREPARE_GLOBAL_INSTANTIATION` werden die Kataloginformationen aller Datenbankobjekte in die Redo Log-Dateien geschrieben. Diese Informationen sind für die Propagation in der Destination Database notwendig.

```
SQL> BEGIN
  2  DBMS_CAPTURE_ADM.PREPARE_GLOBAL_INSTANTIATION();
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

4. Ermitteln Sie die aktuelle System Change Number (SCN) auf der Source Database.

```
SQL> DECLARE
  2  current_scn NUMBER;
  3  BEGIN
  4  current_scn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER;
  5  DBMS_OUTPUT.PUT_LINE(current_scn);
  6  END;
  7  /
608491
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

5. Archivieren Sie die aktuelle Online Redo Log-Datei.

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
System wurde geändert.
```

6. Erstellen Sie die Destination Database mit RMAN. Erzeugen Sie eine Auxiliary Database und verwenden Sie den RMAN-Befehl `DUPLICATE DATABASE`.

7. Erstellen Sie in der Destination Database eine Destination-Queue.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.SET_UP_QUEUE( queue_table=>'streams_queue_table',
   queue_name=>'streams_queue');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

8. Erzeugen Sie einen Datenbank-Link von der Source Database zur Destination Database.

```
SQL> CREATE DATABASE LINK
  2  repl_link
  3  CONNECT TO streams_adm
  4  IDENTIFIED BY streams_adm
  5  USING 'komp10g3';
Datenbank-Link wurde angelegt.
```

9. Definieren Sie die Propagation zur Destination Database in der Source Database.

```
SQL> BEGIN
  2  DBMS_STREAMS_ADM.ADD_GLOBAL_PROPAGATION_RULES(
    streams_name=>'replication',
    source_queue_name=> 'streams_adm.streams_queue',
    destination_queue_name=>'streams_adm.streams_queue@repl_link',
    include_dml=>true, include_ddl=>true,
    source_database=>'komp10g.world', inclusion_rule=>true);
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

10. Erstellen Sie in der Destination Database einen Apply-Prozess.

```
SQL> BEGIN
  2  DBMS_APPLY_ADM.CREATE_APPLY( queue_name=>'streams_queue',
    apply_name=>'repl_apply', apply_user=>'streams_adm',
    apply_captured=>true, source_database=>'komp10g2');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

11. Setzen Sie die System Change Number (SCN) für die Instantiation auf der Destination Database.

```
SQL> BEGIN
  2  DBMS_APPLY_ADM.SET_GLOBAL_INSTANTIATION_SCN(
    source_database_name=>'komp10g2.world', instantiation_scn=>608491,
    recursive=>true);
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

12. Aktivieren Sie zum Schluss den Capture-Prozess und den Apply-Prozess.

```
SQL> CONNECT streams_adm/streams_adm@komp10g2
Connect durchgeführt.
SQL> BEGIN
  2  DBMS_CAPTURE_ADM.START_CAPTURE( capture_name=>'replication');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
SQL> CONNECT streams_adm/streams_adm@komp10g3
Connect durchgeführt.
SQL> BEGIN
  2  DBMS_APPLY_ADM.START_APPLY( apply_name=>'repl_apply');
  3  END;
  4  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Damit ist die Instantiation einer Oracle Streams-Replication-Architektur abgeschlossen und alle erforderlichen Prozesse sind gestartet. Zur Bewertung der neuen Replikationstechnologie lässt sich Positives und Negatives anmerken:

- Erfreulich ist die Tatsache, dass Oracle Replication jetzt auf einer modernen Technologie unter Verwendung von Oracle Streams basiert. Auch das asynchrone Verhalten und die Verwendung von Advanced Queuing als Übertragungsmethode tragen entscheidend zur Erhöhung von Zuverlässigkeit und Datenkonsistenz bei.
- Negativ ist anzumerken, dass die Prozesse des Herstellens und der Überwachung einer Replikationsumgebung sehr aufwendig sind. Hier fehlen die entsprechenden Werkzeuge mit grafischer Oberfläche. Eine Mindestanforderung ist die Funktionalität, wie man sie im Enterprise Manager für Advanced Replication immer noch vorfindet. Eine Topologie-Übersicht wäre ebenfalls wünschenswert. Es ist zu erwarten, dass Oracle hier noch nachbessert. Bis dahin wird man sich wohl mit selbst gebastelten Skripten helfen müssen.

Überwachung von Streams Replication

Auch für die Überwachung von Oracle Streams Replication fehlen geeignete Werkzeuge. Hier ist man wieder auf Views des Datenbankkatalogs angewiesen. Da die Architektur vorwiegend auf Streams-Technologie basiert, bedeutet Überwachung von Replication eine Überwachung der Prozesse Capture, Propagation (Queues) und Apply.

Zusätzlich ist es erforderlich, die Instantiation zu kontrollieren. Mit der SQL-Abfrage in Listing 3.4 können Sie feststellen, welche Datenbankobjekte für die Instantiation vorbereitet wurden.

Listing 3.4: Abfrage von Objekten, die für die Instantiation vorbereitet wurden

```
SQL> SELECT table_owner, table_name, scn, timestamp
  2  FROM dba_capture_prepared_tables;
TABLE_OWNER  TABLE_NAME                          SCN TIMESTAMP
------------ ------------------------------ ---------- -------------------
KOMP10G      BONUS                          608399 12.05.2004 13:21:49
KOMP10G      SALGRADE                       608400 12.05.2004 13:21:49
. . .
```

Die Abfrage in Listing 3.5 liefert Informationen darüber, welche Instantiation SCN die einzelnen Objekte besitzen.

Listing 3.5: Abfrage der Instantiation SCN

```
SQL> SELECT source_object_owner, source_object_name, instantiation_scn
  2  FROM dba_apply_instantiated_objects;
SOURCE_OBJECT_OWNER SOURCE_OBJECT_NAME INSTANTIATION_SCN
------------------- -------------------------- --------
```

```
KOMP10G          EMPLOYEE                  608491
KOMP10G          EMP_DEPT                  608491
. . .
```

Migration von Advanced Replication

Für die Migration von Advanced Replication nach Streams Replication stellt Oracle die Prozedur `DBMS_REPCAT.STREAMS_MIGRATION` zur Verfügung. Sie können eine Liste von Replikationsgruppen als Parameter übergeben. Die Prozedur generiert auf Basis dieser Informationen ein SQL-Skript, das dann in jeder Master-Datenbank von Advanced Replication ausgeführt wird.

Im Skript befinden sich Hinweise zu Objekten und Prozessen, die nicht nach Streams Replication migriert werden können. Sie können das generierte Skript nach Ihren Belangen anpassen.

Listing 3.6: Ein Skript mit dem Migrationstool erzeugen

```
SQL> DECLARE
  2  groups DBMS_UTILITY.NAME_ARRAY;
  3  BEGIN
  4  groups(1) := 'HR';
  5  groups(2) := 'SALES';
  6  DBMS_REPCAT.STREAMS_MIGRATION( gnames=>groups,
file_location=>'/home/oracle', filename=>'streams_mig.sql');
  7  END;
  8  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Die Prozedur verwendet das Paket `UTL_FILE`*. Binden Sie deshalb das Verzeichnis, das Sie im Parameter* `file_location` *verwenden, in den Initialisierungsparameter* `UTL_FILE_DIR` *ein bzw. erstellen Sie ein Oracle Directory*

:-) TIPP

3.3 Oracle Streams Advanced Queuing

Hinter Oracle Streams Advanced Queuing verbirgt sich das Produkt, das Sie als Advanced Queuing aus früheren Oracle-Versionen kennen. Der neue Name wurde nicht nur aus Marketinggründen gewählt, er demonstriert auch die Integration von Advanced Queuing in Oracle Streams.

Advanced Queuing ist also einerseits ein Bestandteil und andererseits eine wesentliche Komponente der Streams-Technologie. Ohne Advanced Queuing ist Oracle Streams nicht funktionsfähig.

Oracle 10*g* stellt die folgenden Erweiterungen und Verbesserungen für Advanced Queuing bereit:

- neue Pakete `DBMS_AQ` und `DBMS_AQADM`

- Streams Messaging Client
- neue AQ-Typen für Array-Verarbeitung
- Vereinfachung der Konfiguration
- neue Views für Buffered Queues

Die Pakete *DBMS_AQ* und *DBMS_AQADM* wurden in Oracle 10*g* komplett überarbeitet. Sie bieten jetzt u.a. eine Unterstützung für Buffered Queues an.

Mit dem Streams Messaging Client ist es möglich, Nachrichten aus der Queue SYS.ANYDATA basierend auf Regeln auszulesen. Sie können einen Messaging Client erstellen, indem Sie DEQUEUE als Wert für den Parameter STREAMS_TYPE im Paket DBMS_STREAMS_ADM verwenden. Neu ist der AQ-Typ Message_Properties_T_Array.

Zur Vereinfachung der Konfiguration trägt das neue Paket DBMS_STREAMS_MESSAGING bei. Damit ist es einfacher, Nachrichten in Queues einzustellen bzw. aus Queues auszulesen, die mit der Prozedur DBMS_STREAMS_ADM.SET_UP_QUEUE erstellt wurden. Die Queue muss daher vom Typ SYS.ANYDATA sein.

INFO

SYS.ANYDATA *ist ein neuer Queue-Typ in Oracle 10*g*, der für Oracle Streams eingeführt wurde. Damit können Nachrichten verschiedenen Typs in einer Queue verwendet werden. Ausnahmen bilden die folgenden Datentypen, die nicht in* SYS.ANYDATA *eingestellt werden können:*

- *CLOB*
- *BLOB*
- *NCLOB*

Auch im Bereich Message Notifications gibt es Verbesserungen. Mit der neuen Prozedur SET_MESSAGE_NOTIFICATION im Paket DBMS_STREAMS_ADM können Sie Benachrichtigungen schicken lassen, wenn Nachrichten zum Auslesen zur Verfügung stehen. Die Benachrichtigung kann an eine E-Mail-Adresse, einen URL oder eine PL/SQL-Prozedur geschickt werden.

Die Architektur von Oracle Streams AQ

Message Queuing ist eine moderne Architektur, die insbesondere mit der Ausbreitung von Internet-Architekturen an Bedeutung gewonnen hat. So kommunizieren webbasierte Anwendungen vorwiegend asynchron über Message Queuing-Systeme miteinander.

Oracle Streams Advanced Queuing unterstützt eine Vielzahl von Architekturen und Protokollen, wie z.B. Oracle Net und HTTPS. AQ kann in alle

denkbaren Applikationsstrukturen integriert werden, was auch als *Integrated Application Environment* bezeichnet wird. In Abbildung 3.11 finden Sie eine Darstellung der Architekturmöglichkeiten.

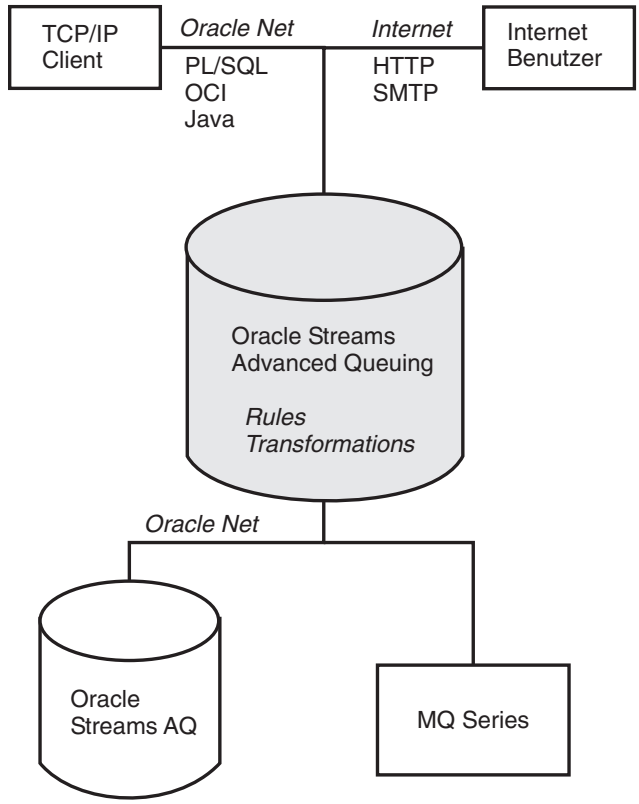


Abbildung 3.11: Die Architektur von Oracle Streams AQ

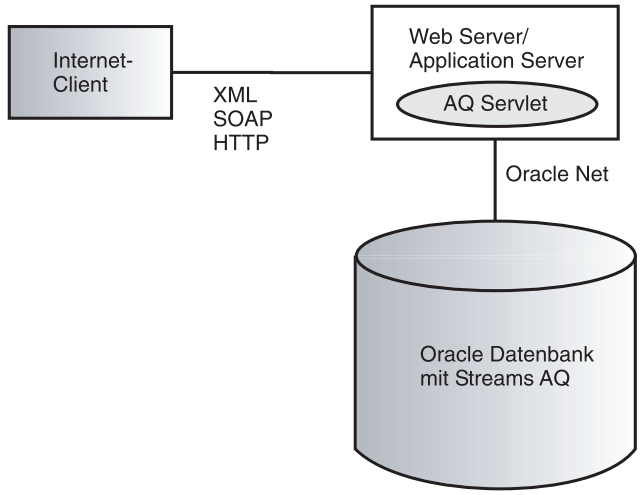
Alle Nachrichten werden in einer Oracle-Datenbank gespeichert, was Vorteile gegenüber anderen Messaging-Systemen bietet. So besteht automatisch eine Transaktionssicherheit im Prozess der Nachrichtenübermittlung und es können sehr große Datenmengen in den Queues vorgehalten werden. Die Menge der vorgehaltenen Daten ist durch die Größe der Oracle-Datenbank begrenzt, die ja bekanntlich bereits mehrere hundert bzw. tausend Terabyte speichern kann.

Über einen Messaging Gateway Agent können mit wenig Aufwand Messaging-Systeme anderer Hersteller eingebunden werden.

AQ unterstützt den Einsatz für Internet- und Intranet-Applikationen durch vielfältige Protokolle. In Abbildung 3.12 finden Sie ein Beispiel für einen Einsatz von AQ im Internet-Umfeld.

Der Internet-Client sendet und empfängt Nachrichten in Form von XML-Dokumenten oder als SOAP Envelope. Diese werden durch den Webserver/Application Server verarbeitet. Dazu stellt Oracle ein Servlet bereit. Das AQ-Servlet stellt eine einfache Kommunikation mit den Queues zur Verfügung und kann Internet-Standard-Protokolle einfach verarbeiten.

Abbildung 3.12: Eine Internet-Architektur mit Advanced Queuing



Der Absender einer Nachricht wird *Producer* genannt, wogegen der Empfänger *Consumer* heißt. Ein Consumer kann, muss aber nicht *Subscriber* einer Queue sein. Ein Subscriber ist ein Agent, der vom AQ-Administrator autorisiert ist, Nachrichten aus der Queue auszulesen. Häufig taucht im Zusammenhang mit AQ auch der Begriff *Recipient* auf. Recipient dient als Oberbegriff für den Empfänger einer Nachricht.

Advanced Queuing unterstützt *Multiconsumer Queues*. Theoretisch müsste ein Producer dieselbe Nachricht mehrfach, für jeden Recipient einstellen. Bei Verwendung von Multiconsumer Queues kann das vermieden werden. Die Nachricht wird dann nur einmal eingestellt und verbleibt so lange in der Queue, bis sie von allen Recipients ausgelesen wurde.

Nachrichten, die von mehreren Recipients ausgelesen werden können, werden auch *Publish/Subcsribe-Nachrichten* genannt. Für die Nachrichtenübermittlung mit Publish/Subscribe stehen zwei Methoden zur Verfügung, *Broadcast* und *Multicast*.

Broadcast ist mit dem Ausstrahlen von Fernseh-Sendern zu vergleichen, bei dem der Sender nicht weiß, wer das Programm konsumiert. Die Empfänger der Nachrichten sind Subscriber von Multiconsumer Queues.

Dagegen kann Multicast mit der Herausgabe eines Magazins verglichen werden, für das der Herausgeber die Abonnenten kennt. In AQ sendet der Publisher Nachrichten gezielt zu verschiedenen Recipients, die Subscriber sein können, aber nicht müssen.

Anwendungen verwenden häufig verschiedene Datenformate. In Advanced Queuing besteht die Möglichkeit einer *Transformation*. Eine Transformation ist eine SQL-Funktion, die einen Quelldatentyp als Parameter verarbeitet und den Zieldatentyp zurückliefert. Sie können eine Transformation beim Einstellen oder beim Auslesen von Nachrichten durchführen.

Sie können Oracle Streams AQ mit den folgenden Werkzeugen bearbeiten:

- PL/SQL unter Benutzung der Pakete `DBMS_AQ`, `DBMS_AQADM`, `DBMS_AQELM`
- C und C++ unter Verwendung des Oracle-OCI-Interface
- Java Messaging Service mit dem Paket `oracle.jms`
- Visual Basic mit Oracle Objects for OLE
- Internet Access mit HTTP

Eine Umgebung für AQ herstellen

Dieser Abschnitt beschreibt, wie Sie eine funktionsfähige Umgebung für Advanced Queuing herstellen können.

Überzeugen Sie sich, bevor Sie mit dem Einrichten beginnen, dass der Initialisierungsparameter `AQ_TM_PROCESSES` auf einen Wert größer null gesetzt ist. Er legt die Anzahl der Queue Monitor-Prozesse fest. Steht er auf null, dann wird kein Queue Monitor-Prozess gestartet. Dies hat zur Folge, dass z.B. Nachrichten nicht aus Multiconsumer-Queues gelöscht werden, wenn sie von allen Subscribern gelesen wurden.

Zuerst müssen Sie einen AQ-Administrator einrichten. Der Administrator erhält Privilegien zur Verwaltung der AQ-Architektur. Im Beispiel erhält der AQ-Administrator den Namen `AQADM` und der Benutzer von AQ den Namen `AQ`.

Obwohl der Datenbankadministrator alle erforderlichen Rechte für AQ besitzt, ist in der Praxis der DBA in der Regel nicht identisch mit dem AQ-Administrator. Es ist daher sinnvoll, einen AQ-Administrator anzulegen.

:-) TIPP

```
SQL> CREATE USER aqadm IDENTIFIED BY aqadm;
Benutzer wurde angelegt.
SQL> GRANT CONNECT, RESOURCE TO aqadm;
Benutzerzugriff (Grant) wurde erteilt.
```

Listing 3.7: AQ-Administrator und AQ-Benutzer anlegen

```
SQL> GRANT EXECUTE ON dbms_aqadm TO aqadm;
Benutzerzugriff (Grant) wurde erteilt.
SQL> GRANT AQ_ADMINISTRATOR_ROLE TO aqadm;
Benutzerzugriff (Grant) wurde erteilt.
SQL> CREATE USER aq IDENTIFIED BY aq;
Benutzer wurde angelegt.
SQL> GRANT CONNECT, RESOURCE TO aq;
Benutzerzugriff (Grant) wurde erteilt.
SQL> GRANT EXECUTE ON dbms_aq TO aq;
Benutzerzugriff (Grant) wurde erteilt.
```

Queue-Tabellen können mit verschiedenen Datentypen angelegt werden. Prüfen Sie vor dem Anlegen der Queue-Tabelle, welche Art von Daten transportiert werden sollen. Im Beispiel wird ein Object Type verwendet. Sie können individuell festlegen, welche Felder und Datentypen im Objekttyp enthalten sein sollen.

Listing 3.8: Eine Queue vom Typ `Object Type` erstellen

```
SQL> CREATE TYPE aq.message_typ AS OBJECT (
  2  subject    VARCHAR2(20),
  3  contents   VARCHAR2(60));
  4  /
Typ wurde erstellt.
SQL> EXEC DBMS_AQADM.CREATE_QUEUE_TABLE(
queue_table=>'aq.queue_tab_obj_table',
queue_payload_type=>'aq.message_typ');
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
SQL> EXEC DBMS_AQADM.CREATE_QUEUE( queue_name=>'aq.queue_tab_obj',
queue_table=>'aq.queue_tab_obj_table');
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Die Queue besitzt nach dem Anlegen den Status »gestoppt«. Starten Sie die Queue mit der Prozedur `START_QUEUE`.

```
SQL> EXEC DBMS_AQADM.START_QUEUE( queue_name=>'aq.queue_tab_obj');
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Verifizieren Sie mit dem View `DBA_QUEUES`, dass die Queue existiert und gestartet wurde. Ist die Queue gestartet, dann müssen die Spalten `ENQUEUE_ENABLED` und `DEQUEUE_ENABLED` den Wert `YES` besitzen. Die Queue ist damit bereit zum Einstellen und Auslesen von Nachrichten.

Listing 3.9: Eine Queue verifizieren

```
SQL>  SELECT name, queue_type, enqueue_enabled, dequeue_enabled
  2  FROM dba_queues
  3  WHERE owner = 'AQ';
NAME                         QUEUE_TYPE      ENQUEUE DEQUE
---------------------------- ----------------- ------- ---
AQ$_QUEUE_TAB_OBJ_TABLE_E  EXCEPTION_QUEUE   NO       NO
QUEUE_TAB_OBJ              NORMAL_QUEUE      YES     YES
```

Wie Sie sicher bemerkt haben, wurde neben der Queue mit dem Namen QUEUE_TAB_OBJ *eine weitere mit dem Namen* AQ$_QUEUE_TAB_OBJ_TABLE_E *angelegt. Dabei handelt es sich um eine Exception-Queue. Sie wird standardmäßig mit jeder Queue angelegt. In einer Exception-Queue werden alle Nachrichten gespeichert, die nicht innerhalb der vorgegebenen Bedingungen oder des vorgegebenen Zeitfensters verarbeitet werden können. Die Spalten* ENQUEUE_ENABLED *und* DEQUEUE_ENABLED *sind für Exception-Queues immer auf* NO *gesetzt, da in sie nicht mit Interfaceoperationen geschrieben werden kann.*

Sie können sich die Queue und die Queue-Tabelle auch mit dem Enterprise Manager ansehen. Beachten Sie, dass Oracle Streams nicht im neuen webbasierenden Enterprise Manager integriert sind. Sie müssen also den Java-Client des EM aufrufen. Neben allgemeinen Informationen zu Queue und Queue Table finden Sie auch Statistiken über die Queue.

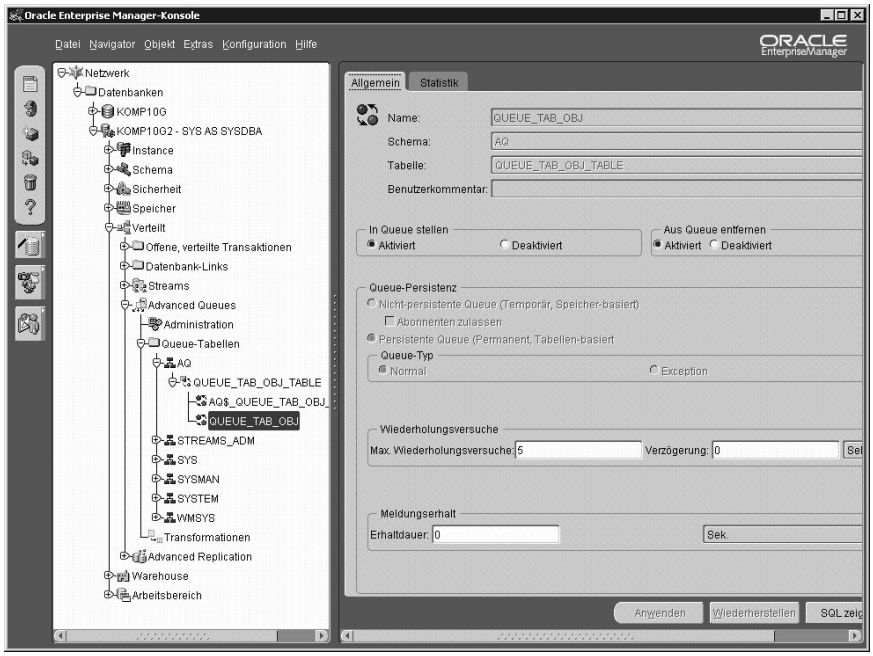


Abbildung 3.13: Verwaltung von Advanced Queuing im Enterprise Manager

Sie müssen sich beim Anlegen einer Queue Table entscheiden, ob es sich um eine Multiconsumer Queue oder eine Singleconsumer Queue handeln soll. Wenn Sie nichts spezifizieren, wird standardmäßig eine Singleconsumer Queue angelegt.

Die soeben erstellte Singleconsumer Queue ist gestartet und damit bereit, Nachrichten zu empfangen. Wie bereits erwähnt, können Sie verschiedene Sprachen benutzen, um Nachrichten einzustellen und auszulesen. Dabei gibt es keine Einschränkungen. Sie können z.B. eine Nachricht mit PL/SQL ein-

stellen und mit einem Java-Interface auslesen. Im vorliegenden Beispiel verwenden wir PL/SQL.

Die PL/SQL-Prozedur in Listing 3.10 stellt eine Nachricht vom Typ Object Type in die Queue ein. Verwenden Sie den Benutzer AQ, den wir als AQ-Benutzer angelegt haben.

Listing 3.10: Eine Nachricht mit PL/SQL einstellen

```
SQL> CONNECT aq/aq@komp10g2
Connect durchgeführt.
SQL> DECLARE
  2  enqueue_options     DBMS_AQ.ENQUEUE_OPTIONS_T;
  3  message_properties  DBMS_AQ.MESSAGE_PROPERTIES_T;
  4  message_handle      RAW(16);
  5  message             AQ.MESSAGE_TYP;
  6  BEGIN
  7  message := message_typ('First AQ message','Enqueued with PL/SQL');
  8  DBMS_AQ.ENQUEUE(queue_name=>'queue_tab_obj',
enqueue_options=>enqueue_options, message_properties=>message_properties,
payload=>message, msgid=>message_handle);
  9  COMMIT;
 10  END;
 11  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Die einfachste Methode, die Anzahl von Nachrichten in einer Queue zu ermitteln, ist ein Zählen von Sätzen in der Queue Table.

```
SQL> SELECT COUNT(*)
  2  FROM aq.queue_tab_obj_table
  3  WHERE q_name = 'QUEUE_TAB_OBJ';
  COUNT(*)
----------
         1
```

Sie können gespeicherte Nachrichten auch im Enterprise Manager ansehen. Klicken Sie dazu mit der rechten Maustaste auf die Queue und wählen Sie im Kontext-Menü MELDUNGEN ANZEIGEN. Es öffnet sich ein Fenster, in dem Sie alle Nachrichten sehen, die in der Queue gespeichert sind.

Das Auslesen der Nachricht erfolgt analog zum Einstellen mit einer PL/SQL-Prozedur, die Sie in Listing 3.11 finden. Da es sich um eine Singleconsumer Queue handelt, wird die Nachricht nach dem Auslesen gelöscht. Alternativ können Sie die Option einstellen, dass die Nachricht nach dem Lesen in der Queue verbleiben soll. Das erfolgt über den Parameter `DEQUEUE_MODE` in den Dequeue Options.

Abbildung 3.14: Nachrichten im Enterprise Manager anzeigen

Listing 3.11: Eine Nachricht mit PL/SQL auslesen

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2  dequeue_options     DBMS_AQ.DEQUEUE_OPTIONS_T;
  3  message_properties  DBMS_AQ.MESSAGE_PROPERTIES_T;
  4  message_handle      RAW(16);
  5  message             AQ.MESSAGE_TYP;
  6  BEGIN
  7  DBMS_AQ.DEQUEUE(queue_name=>'queue_tab_obj',
dequeue_options=>dequeue_options, message_properties=>message_properties,
payload=>message, msgid=>message_handle);
  8  DBMS_OUTPUT.PUT_LINE(message.subject||' : '||message.contents);
  9  END;
 10  /
First AQ message : Enqueued with PL/SQL
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Für das Einstellen und Auslesen von Nachrichten aus einer Queue sind Privilegien erforderlich. Der Besitzer der Queue erhält dieses Recht standardmäßig. Wenn jedoch andere Benutzer mit der Queue arbeiten sollen, dann müssen Sie diesen entsprechende Privilegien zuweisen. Es empfiehlt sich, den Benutzern einzeln Rechte an einer Queue zu geben. Verwenden Sie hierfür die Prozedur `GRANT_ENQUEUE_PRIVILEGE` *wie im folgenden Beispiel.*

```
DBMS_AQADM.GRANT_QUEUE_PRIVILEGE(privilege=>'DEQUEUE', queue_name=>'AQ.QUEUE_
   TAB_OBJ', grantee=>'AQ_USER', grant_option=>FALSE);
COMMIT;
```

Alternativ können Sie die Privilegien `ENQUEUE ANY QUEUE` *und* `DEQUEUE ANY QUEUE` *vergeben. Seien Sie jedoch vorsichtig mit der Vergabe dieser Privilegien.*

Im folgenden Beispiel soll eine Multiconsumer-Queue angelegt werden. Wie Sie bereits wissen, muss der Queue-Typ beim Erstellen der Queue Table festgelegt werden.

Listing 3.12: Eine Multiconsumer-Queue erstellen

```
SQL> EXEC DBMS_AQADM.CREATE_QUEUE_TABLE( queue_table=>'aq.mcons_queue_table',
    multiple_consumers=>true, queue_payload_type=>'aq.message_typ');
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
SQL> EXEC DBMS_AQADM.CREATE_QUEUE( queue_name=>'aq.mcons_queue',
queue_table=>'aq.mcons_queue_table');
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
SQL> EXEC DBMS_AQADM.START_QUEUE( queue_name=>'aq.mcons_queue');
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Für eine Multiconsumer Queue müssen Subscriber definiert werden. Nachrichten werden genau dann aus der Queue gelöscht, wenn alle Subscriber die Nachricht abgeholt haben. In Listing 3.13 sehen Sie, wie mit PL/SQL die Subscriber `sub1` und `sub2` zu einer Queue hinzugefügt werden.

Listing 3.13: Subscriber zu einer Multiconsumer-Queue hinzufügen

```
SQL> DECLARE
  2  subscriber    sys.aq$_agent;
  3  BEGIN
  4  subscriber := sys.aq$_agent( 'sub1', 'aq.mcons_queue', null);
  5  DBMS_AQADM.ADD_SUBSCRIBER( queue_name=>'aq.mcons_queue',
subscriber=>subscriber);
  6  subscriber := sys.aq$_agent( 'sub2', 'aq.mcons_queue', null);
  7  DBMS_AQADM.ADD_SUBSCRIBER( queue_name=>'aq.mcons_queue',
subscriber=>subscriber);
  8  END;
  9  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

In Listing 3.14 wird eine Nachricht in die Multiconsumer-Queue eingefügt. Die Nachricht ist für die Subscriber der Queue bestimmt.

Listing 3.14: Eine Nachricht in eine Multiconsumer-Queue für Subscriber einstellen

```
SQL> DECLARE
  2  enqueue_options    dbms_aq.enqueue_options_t;
  3  message_properties dbms_aq.message_properties_t;
  4  recipients         dbms_aq.aq$_recipient_list_t;
  5  message_handle     RAW(16);
  6  message            message_typ;
  7  BEGIN
  8  message := message_typ('Multi Cons Message',
'Message is queued for subscribers');
  9  DBMS_AQ.ENQUEUE( queue_name=>'aq.mcons_queue',
enqueue_options=>enqueue_options, message_properties=>message_properties,
payload=>message, msgid=>message_handle);
 10  COMMIT;
 11  END;
 12  /
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

In Listing 3.15 werden alle vorhandenen Nachrichten für einen Subscriber aus der Multiconsumer Queue ausgelesen. Sind keine Nachrichten mehr vorhanden, dann wird eine Exception erzeugt und das Programm beendet. Dabei wird die Option `no_wait` verwendet, das heißt, das Programm wartet nicht, wenn keine Nachricht in der Queue gefunden wird.

Listing 3.15: Nachrichten aus einer Multiconsumer-Queue durch Subscriber auslesen

```
SQL> DECLARE
  2  dequeue_options     dbms_aq.dequeue_options_t;
  3  message_properties  dbms_aq.message_properties_t;
  4  message_handle      RAW(16);
  5  message             aq.message_typ;
  6  empty               exception;
  7  pragma              exception_init( empty, -25228);
  8  BEGIN
  9    dequeue_options.wait := dbms_aq.no_wait;
 10    BEGIN
 11    dequeue_options.consumer_name := 'sub1';
 12    dequeue_options.navigation := DBMS_AQ.FIRST_MESSAGE;
 13    LOOP
 14      DBMS_AQ.DEQUEUE( queue_name=>'aq.mcons_queue',
dequeue_options=>dequeue_options,
message_properties=>message_properties,
payload=>message, msgid=>message_handle);
 15      DBMS_OUTPUT.PUT_LINE( message.subject || ' : ' || message.contents);
 16      dequeue_options.navigation := DBMS_AQ.NEXT_MESSAGE;
 17    END LOOP;
 18    EXCEPTION WHEN empty THEN
 19      DBMS_OUTPUT.PUT_LINE('Keine Nachrichten fuer SUB1');
 20    COMMIT;
 21    END;
 22  END;
 23  /
Multi Cons Message : Message is queued for subscribers
Keine Nachrichten fuer SUB1
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Wenn Sie die Nachricht für den Subscriber `SUB1` ausgelesen haben, dann werden Sie feststellen, dass die Nachricht noch in der Queue verbleibt, bis auch der Subscriber `SUB2` die Nachricht ausgelesen hat. Im Enterprise Manager ist die Meldung für `SUB1` als `PROCESSED` markiert, die für `SUB2` besitzt den Status `READY`.

Auch wenn im Meldungsfenster des Enterprise Manager zwei Zeilen angezeigt werden, ist die Nachricht physisch nur einmal in der Queue vorhanden.

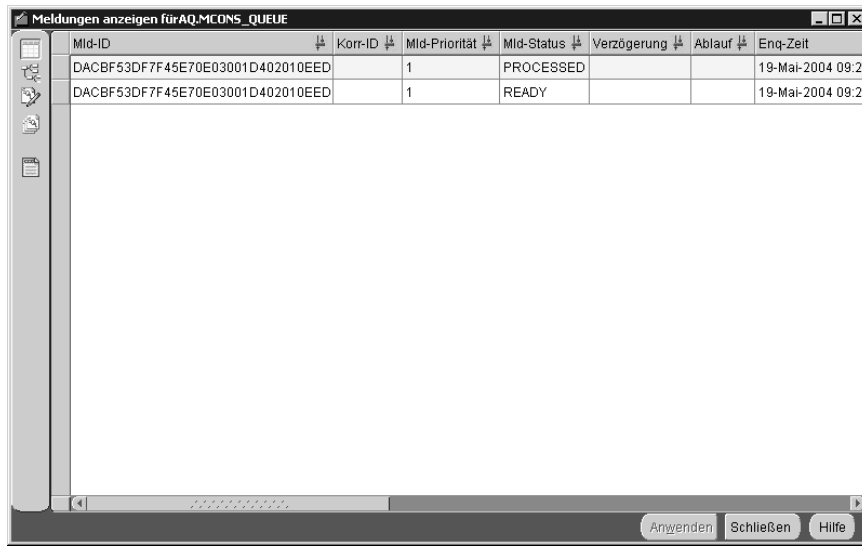


Abbildung 3.15: Nachrichten einer Multiconsumer-Queue im Enterprise Manager anzeigen

:-) TIPP

Sie können eine Subscriber-Liste überschreiben, indem Sie Recipients für die Nachricht angeben. Recipients sind Eigenschaften einer Nachricht, nicht einer Queue und können für jede Nachricht individuell gewählt werden. Recipients können, müssen aber nicht Subscriber der Queue sein. Die Definition erfolgt beim Einstellen der Nachricht in der folgenden Form:

```
recipients    dbms_aq.aq$_recipient_list_t;
. . .
recipients(1) := aq$_agent('SUB1', NULL, NULL);
message_properties.recipient_list := recipients;
```

i INFO

Die Nachrichten verschwinden nicht sofort aus der Queue, wenn sie von allen Empfängern ausgelesen wurden. Es kann unter Umständen einige Sekunden dauern, bis die Queue bereinigt ist. Das Bereinigen von Multiconsumer Queues erfolgt durch einen Hintergrundprozess, der asynchron arbeitet und die Queues in regelmäßigen Abständen prüft.

Sie können den Inhalt von Nachrichten von einem Oracle-Datentyp in einen anderen umwandeln. Dieser Prozess heißt *Transformation* und wird von einer SQL-Funktion ausgeführt. Die Funktion verarbeitet den Quelltyp als Parameter und liefert den Zieltyp zurück. Sie können die Transformation beim Einstellen, Auslesen oder der Propagation von Nachrichten definieren.

Im folgenden Beispiel erfolgt eine Transformation der Nachrichten beim Auslesen. Die Nachricht soll an eine Internet-Applikation gesendet werden und deshalb beim Empfänger im XML-Format ankommen.

Eine Nachricht beim Auslesen transformieren

1. Erstellen Sie eine Funktion für die Transformation des Datentyps. Der Funktionsparameter muss den Datentyp der Nachricht besitzen.

```
SQL> CREATE OR REPLACE FUNCTION convert_to_xml( input message_typ)
  2  RETURN XMLType AS
  3  output XMLType;
  4  BEGIN
  5  SELECT SYS_XMLGEN(input) INTO output FROM dual;
  6  RETURN output;
  7  END;
  8  /
Funktion wurde erstellt.
```

2. Erstellen Sie eine Transformation unter Verwendung des Paketes `DBMS_TRANSFORM`. Geben Sie als Transformation die soeben erstellte Funktion `CONVERT_TO_XML` an.

```
SQL> EXEC DBMS_TRANSFORM.CREATE_TRANSFORMATION( schema=>'aq',
    name=>'xml_transformation', from_schema=>'aq',
    from_type=>'message_typ', to_schema=>'sys',
    to_type=>'XMLType',
    transformation=>'aq.convert_to_xml( source.user_data)');
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

3. Lesen Sie die Nachricht aus der Queue aus. Beachten Sie, dass die Nachricht nicht mehr vom Typ `message_typ`, sondern vom Typ `XMLType` ist, wenn Sie die Transformation in den Message Properties spezifizieren.

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2  dequeue_options     DBMS_AQ.DEQUEUE_OPTIONS_T;
  3  message_properties  DBMS_AQ.MESSAGE_PROPERTIES_T;
  4  message_handle      RAW(16);
  5  message             XMLType;
  6  BEGIN
  7  dequeue_options.transformation := 'AQ.XML_TRANSFORMATION';
  8  DBMS_AQ.DEQUEUE(queue_name=>'queue_tab_obj',
    dequeue_options=>dequeue_options,
    message_properties=>message_properties, payload=>message,
    msgid=>message_handle);
  9  DBMS_OUTPUT.PUT_LINE(message.GETSTRINGVAL());
 10  END;
 11  /
<?xml version="1.0"?>
<ROW>
 <SUBJECT>Message 2</SUBJECT>
 <CONTENTS>Message for XML transformation</CONTENTS>
</ROW>
PL/SQL-Prozedur wurde erfolgreich abgeschlossen.
```

Die Verwaltung von AQ

Für die Verwaltung und Überwachung von Oracle Streams Advanced Queuing stehen der Enterprise Manager (Java-Konsole) und eine Reihe von Views aus dem Datenbankkatalog zur Verfügung. Eine Übersicht der vorhandenen Queues können Sie mit der SQL-Abfrage in Listing 3.16 erstellen. Aus den Spalten `ENQUEUE_ENABLED` und `DEQUEUE_ENABLED` ist ersichtlich, ob die Queue gestartet oder gestoppt ist. Exception-Queues sind immer gestoppt, da in sie nie direkt Nachrichten eingestellt werden können.

Listing 3.16: Übersicht der vorhandenen Queues

```
SQL> SELECT owner, name,
  2  enqueue_enabled, dequeue_enabled FROM dba_queues
  3  WHERE OWNER NOT IN ('SYS', 'SYSTEM', 'WMSYS', 'SYSMAN');
OWNER        NAME                        ENQUEUE DEQUEUE
------------ --------------------------- ------- -------
AQ           AQ$_QUEUE_TAB_OBJ_TABLE_E    NO      NO
AQ           QUEUE_TAB_OBJ                YES     YES
STREAMS_ADM  AQ$_STREAMS_QUEUE_TABLE_E    NO      NO
STREAMS_ADM  STREAMS_QUEUE                YES     YES
AQ           AQ$_MCONS_QUEUE_TABLE_E      NO      NO
AQ           MCONS_QUEUE                  YES     YES
```

Mit Hilfe der Abfrage in Listing 3.17 erhalten Sie eine Übersicht der vorhandenen Queue Tables. Sie können auch erkennen, ob es sich um eine Singleconsumer- oder um eine Multiconsumer-Queue handelt.

Listing 3.17: Queue Tables abfragen

```
SQL> SELECT owner, queue_table, type, recipients
  2  FROM dba_queue_tables
  3  WHERE owner NOT IN  ('SYS', 'SYSTEM', 'WMSYS', 'SYSMAN');
OWNER        QUEUE_TABLE            TYPE    RECIPIEN
------------ ---------------------- ------- --------
AQ           MCONS_QUEUE_TABLE      OBJECT  MULTIPLE
AQ           QUEUE_TAB_OBJ_TABLE    OBJECT  SINGLE
STREAMS_ADM  STREAMS_QUEUE_TABLE    OBJECT  MULTIPLE
```

Eine Statistik und Aussagen über Performance der einzelnen Queues und Queue Tables erhalten Sie mit dem View `V$AQ`. In der Spalte `READY` finden Sie die Anzahl der Nachrichten, die zur Abholung bereit stehen. Die Spalte `EXPIRED` markiert Nachrichten mit verletzten Abholbedingungen, z.B. Zeitüberschreitung. Schließlich finden Sie noch eine Statistik über totale und durchschnittliche Wartezeit. Darunter ist die Zeit zu verstehen, die Nachrichten auf ihre Abholung warten. Sollen Nachrichten immer sofort ausgelesen werden, dann erhalten Sie eine Statistik über die Performance beim Abholen von Nachrichten.

Listing 3.18: Statistiken der Queue Tables auswerten

```
SQL> SELECT a.owner, a.name, b.waiting,
  2  b.ready, b.expired, b.total_wait, b.average_wait
  3  FROM dba_queues a, v$aq b
  4  WHERE a.qid = b.qid
  5  AND a.owner NOT IN ('SYS', 'SYSTEM', 'WMSYS', 'SYSMAN');
OWNER        NAME                          WAITING      READY    EXPIRED
TOTAL_WAIT AVERAGE_WAIT
------------ ------------------------- ---------- ---------- ----------
---------- ------------
AQ           AQ$_QUEUE_TAB_OBJ_TABLE_E          0          0          0
         0            0
AQ           QUEUE_TAB_OBJ                      0         13          0
      7349   565,307692
STREAMS_ADM  AQ$_STREAMS_QUEUE_TABLE_E          0          0          0
         0            0
STREAMS_ADM  STREAMS_QUEUE                      0          0          0
         0            0
AQ           AQ$_MCONS_QUEUE_TABLE_E            0          0          0
         0            0
AQ           MCONS_QUEUE                        0          0          0
         0            0
```

Dieselbe Statistik erhalten Sie im Enterprise Manager pro Queue Table. Wählen Sie die entsprechende Queue Table aus und klicken Sie auf das Register STATISTIK.

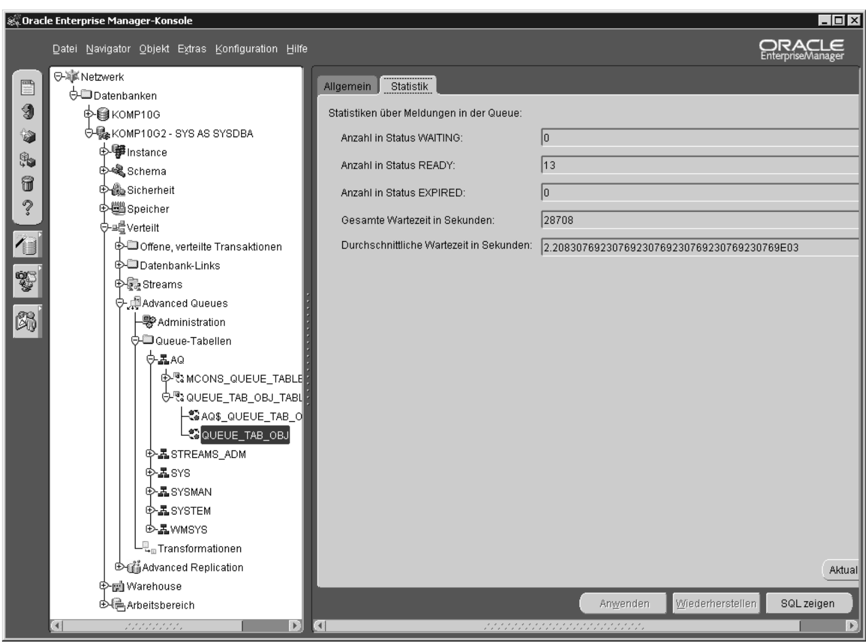


Abbildung 3.16: Die Statistik einer Queue im Enterprise Manager anzeigen