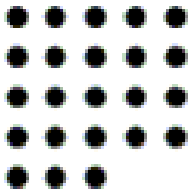


Fachhochschule Köln
University of Applied Sciences Cologne
Campus Gummersbach
Fakultät für Informatik und Ingenieurwissenschaften

Diplomarbeit

zur Erlangung
des Diplomgrades
Diplom-Informatiker (FH)
in der Fachrichtung Informatik

Evaluierung einer Datenbankmigration am Beispiel von Oracle Database und Microsoft SQL Server

Erstprüfer:	Prof. Dr. Heide Faeskorn-Woyke
Zweitprüfer:	Dipl. Ing. Johannes Ahrends
vorgelegt am:	28.08.2007
von:	Erdal Türkeri
	Gelpestr. 42a
	51647 Gummersbach
E-Mail:	Tuerkeri@t-online.de
Matrikelnummer:	11028334

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation und Problemstellung

Zahlreiche Gründe können Unternehmen dazu veranlassen von einem kommerziellen Datenbankmanagementsystem auf das eines anderen Herstellers zu wechseln. Einer der Gründe sind die Kosten. In den letzten Jahren sind bei den kommerziellen Datenbanken Änderungen zu verzeichnen. Dies betrifft sowohl die generelle Leistungsfähigkeit der verschiedenen Systeme als auch die Verschiedenheit der Kosten innerhalb der einzelnen Systeme, ebenso die Kompatibilität der einzelnen Systeme untereinander. Die finanzielle Realität für die meisten Unternehmen ist, dass zu viel ihres Etats durch ihre vorhandene IT-Infrastruktur verbraucht wird und zu wenig für neue IT-Investitionen vorhanden ist. Dementsprechend ist die Senkung der Kosten für die IT-Infrastruktur für viele leitende IT-Manager oberstes Gebot. Datenbanken sind dabei ein lohnenswertes Ziel für geplante Sparmaßnahmen, da hiermit ein erheblicher Aufwand sowohl im Hinblick auf die IT-Arbeitsleistung als auch im Hinblick auf die Personalkosten verbunden ist. Weitere Gründe einer Datenbankmigration können Anforderungen an den Herstellersupport, die Kompatibilität zu anderen Datenbanken oder andere Gründe sein. Hier soll allerdings nicht die genaue Kostenstruktur untersucht werden, sondern die technischen Aspekte einer Umstellung.

Datenbankmigrationen stellen im Hinblick auf die Kosten und die Komplexität eine erhebliche Herausforderung dar. Vorhandene Daten aus einem alten System auf ein neues System eines anderen Herstellers zu übertragen ist sehr aufwendig, weil insbesondere system- bzw. herstellerspezifische Besonderheiten nicht übernommen werden können.

1.2 Zielsetzung und Vorgehensweise

Ziel dieser Diplomarbeit ist anhand von Praxisbeispielen Migrationen zwischen den beiden kommerziellen Datenbankmanagementsystemen Oracle Database und Microsoft SQL Server durchzuführen und die Ergebnisse zu untersuchen.

Dabei werden Migrationswerkzeuge beider Hersteller verwendet, die nach Herstellerangaben eine schnelle, einfache und akkurate Migration von fremden Datenbankmanagementsystemen auf das eigene System unterstützen sollen. Es soll untersucht werden, ob mit diesen Migrationswerkzeugen die Schema- und Datenmigration automatisiert werden kann.

In dieser Diplomarbeit soll zunächst einmal in Kapitel 2 die physische und die logische Datenbankarchitektur von Oracle Database und Microsoft SQL Server gegenübergestellt werden, wobei auch einige theoretische und begriffliche Grundlagen dargelegt werden. Danach werden in Kapitel 3 Datenbankobjekte der beiden Systeme verglichen, wobei unter anderem Datentypzuordnungen und Syntax-Unterschiede aufgeführt werden. In Kapitel 4 werden die unterschiedlichen Sperrkonzepte der beiden Systeme behandelt. In Kapitel 5 wird anhand eines Praxisbeispiels das Ergebnis einer Migration von Oracle zu SQL Server dargelegt. Anschließend wird in Kapitel 6 wiederum anhand eines Praxisbeispiels das Ergebnis einer Migration von SQL Server zu Oracle dargelegt. Schließlich werden in Kapitel 7 die wichtigsten Ergebnisse dieser Diplomarbeit zusammengefasst.

1.3 Danksagung

Ich danke Frau Prof. Dr. Heide Faeskorn-Woyke, durch die ich dieses Diplomarbeitsthema gefunden habe und die mich mit Herrn Ahrends in Kontakt gebracht hat. Herrn Dipl. Ing. Johannes Ahrends danke ich dafür, dass er mir einen Arbeitsplatz bei der Quest Software GmbH zur Verfügung gestellt hat und mir immer mit seinem bemerkenswerten Fachwissen mit Rat und Tat zur Seite stand.

Ebenso bedanke ich mich ausdrücklich bei Herrn Dipl. Inf. Patrick Schwanke, der mich sehr geduldig mit seinen beeindruckenden PL/SQL Kenntnissen unterstützt hat und auch Herrn Dipl. Inf. Thomas Klughardt, der stets ein offenes Ohr für meine Fragen hatte.

Schließlich möchte ich noch allen anderen Quest Mitarbeitern für ihr herzliches Entgegenkommen und ihre Unterstützung danken.

2 Architektur-Vergleich

Oracle Database und Microsoft SQL Server sind relationale Datenbank-managementsysteme. Dabei ist Oracle für fast alle heute verwendeten Plattformen verfügbar wohingegen der SQL Server nur auf Basis von Microsoft Windows Plattformen erhältlich ist. Die Database Engine (Datenbankmodul) des SQL Servers ist der eigentliche Kerndienst, mit dem das Speichern, Verarbeiten und Sichern von Daten realisiert wird. Das Datenbankmodul ist ein Teilbereich des SQL Servers. Daneben gibt es noch viele weitere herstellerabhängige Merkmale, die bei einer Migration beachtet werden müssen. Beispielsweise verläuft die Datenspeicherung, die Datensicherung oder die Transaktionsverwaltung der jeweiligen Systeme grundsätzlich unterschiedlich. Aus diesem Grund sollte zunächst einmal verglichen werden, inwieweit bestimmte Konstruktionen in beiden Systemen vorhanden sind. Daher sollen die unterschiedlichen physischen- und logischen Datenbankarchitekturen nachfolgend kurz erläutert werden. Ferner soll dies auch dazu dienen, einen kleinen Überblick über die unterschiedlichen Nomenklaturen, die in beiden Systemen verwendet werden, zu erlangen.

2.1 Physische Datenbankarchitektur

Oracle-Dateiarten

In einer Oracle-Datenbank werden die folgenden, unverzichtbaren Dateiarten unterschieden[1] [2]:

Kontrolldateien: Diese Dateien enthalten Informationen über die physische Datenbankstruktur, die u.a. für Backup- und Recovery-Aktionen sowie für die Konsistenzsicherung notwendig sind.

Datendateien: Diese Dateien enthalten alle Benutzer- und Metadaten der Datenbank.

[1] Vgl. [Ahrends06] Seite 100
[2] Vgl. [Best05] Kapitel 1, Seite 15

Online-Redolog-Dateien: Diese Dateien enthalten sämtliche Transaktions-protokollinformationen, die zum Wiederherstellen der Datenbank benötigt werden.

SQL Server-Dateiarten

SQL Server verwendet drei Arten von Dateien[3]:

Primäre Datendateien: Enthält die Startinformationen für die Datenbank und zeigt auf die anderen Dateien in der Datenbank.
Sekundäre Datendateien: Sekundäre Datendateien sind optional, benutzerdefiniert und speichern Benutzerdaten.
Protokolldateien: Diese Dateien enthalten sämtliche Transaktionsprotokoll-informationen, die zum Wiederherstellen der Datenbank benötigt werden.

2.2 Logische Datenbankarchitektur

2.2.1 Logische Speichereinheiten

Oracle-Tablespaces

Oracle-Datenbanken sind in logische Speichereinheiten (Tablespaces) unterteilt, die auf der physikalischen Ebene durch Datendateien repräsentiert werden[4]. Durch die Trennung der logischen Architektur und der physikalischen Speicherung wird der gleiche Aufbau der Datenbank auf unterschiedlichen Plattformen ermöglicht. Tablespaces bestehen aus einer oder mehreren Datendateien. Jede Datendatei kann jeweils nur einem Tablespace angehören.
Folgende Tablespaces sollten in einer Oracle 10g-Datenbank nicht fehlen:

SYSTEM: Speichert die Tabellen, die die Kernfunktionalität der Datenbank unterstützen, beispielsweise Data Dictionary-Tabellen. (Obligatorisch)
SYSAUX: Speichert zahlreiche Datenbankkomponenten wie beispielsweise das Enterprise Manager Repository. (Obligatorisch)

[3] Vgl. [MSDN1]
[4] Vgl. [Ahrends06] Seite 101

TEMP: Der temporäre Tablespace wird verwendet, wenn eine SQL-Anweisung ausgeführt wird, bei der temporäre Segmente erstellt werden müssen (z. B. bei einer umfangreichen Sortierung oder einer Indexerstellung)[5].
UNDOTBS: Ein Undo-Tablespace beinhaltet ausschließlich Undo-Segmente, deren Aufgabe es ist, den alten Zustand von Feldinhalten (before images) so lange zu speichern, bis der neue Zustand in der Datenbank festgeschrieben wurde.
USERS: In diesem Tablespace werden permanente Benutzerobjekte und -daten gespeichert.

SQL Server-Dateigruppen

Jede Datendatei wird zu Verwaltungs- und Zuordnungszwecken in einer Dateigruppe gespeichert[6]. Durch die Aufteilung der Daten auf mehrere Dateien und Dateigruppen können diese separat gesichert (Dateigruppensicherung) und wiederhergestellt werden. Jede Datei kann jeweils nur einer Dateigruppe angehören. Es gibt zwei Arten von Dateigruppen:

Primäre Dateigruppe: Enthält die primäre Datendatei und alle anderen Dateien, die nicht explizit einer anderen Dateigruppe zugewiesen sind. Alle Seiten für die Systemtabellen werden in der primären Dateigruppe zugeordnet.
Benutzerdefinierte Dateigruppe: Alle Dateigruppen, die mit Hilfe des FILEGROUP-Schlüsselwortes in der Anweisung CREATE DATABASE oder ALTER DATABASE angegeben werden.

SQL Server-Systemdatenbanken

In SQL Server wird eine Reihe von Datenbanken auf Systemebene verwendet – die Systemdatenbanken, die für den Betrieb einer Serverinstanz von entscheidender Bedeutung sind.
Jede SQL Server-Datenbank enthält die folgenden Systemdatenbanken:

master-Datenbank: Zeichnet alle Informationen auf Systemebene für eine Instanz von SQL Server auf. Dazu gehören Metadaten wie Anmeldekonten, Endpunkte, Verbindungsserver und Systemkonfigurationseinstellungen.

[5] Vgl. [Best05] Kapitel 5, Seite 15
[6] Vgl. [MSOD2]

msdb-Datenbank: Wird von SQL Server-Agent verwendet, um Termine für Warnungen und Aufträge zu planen.

tempdb-Datenbank: Ein Arbeitsbereich zum Speichern von temporären Objekten oder Zwischenresultsets. Die tempdb-Systemdatenbank ist eine globale Ressource, die für alle Benutzer verfügbar ist, die mit der Instanz von SQL Server verbunden sind.

Ressourcendatenbank: Eine schreibgeschützte Datenbank, die alle Systemobjekte enthält, die in SQL Server enthalten sind.

model-Datenbank: Wird als Vorlage für alle Datenbanken verwendet, die für die Instanz von SQL Server erstellt werden.

Schema-Objekte

Abschließend werden in der nachfolgenden Tabelle noch einmal einige Schema-Objekte dieser beiden Datenbanken verglichen. Weitere Schema-Objekte werden in Abschnitt 3.2 besprochen:

Schema-Objekte von Oracle und SQL-Server

Oracle	SQL Server
Datenbank	Datenbank
Schema	Datenbank und Datenbankbesitzer (DBO)
Tablespace	Dateigruppe
Benutzer	Benutzer
Rolle	Gruppe/Rolle
Tabelle	Tabelle
Temporäre Tabellen	Temporäre Tabellen
Cluster	*<nicht unterstützt>*
Column-level check constraint	Column-level check constraint
Column default	Column default
Unique-Schlüssel	Unique-Schlüssel oder Identitätsmerkmal für die Zeile
Primärschlüssel	Primärschlüssel
Fremdschlüssel	Fremdschlüssel
PL/SQL Prozedur	T-SQL Gespeicherte Prozedur

PL/SQL Funktion	T-SQL Funktion
Snapshot	*<nicht unterstützt>*
View	View

Tabelle 1: Schema-Objekte von Oracle und SQL-Server

2.2.2 Instanzen

Oracle-Instanz

Ein Oracle-Datenbank-Server besteht aus einer Oracle-Datenbank und einer Oracle-Instanz[7]. Oracle-Instanzen bestehen aus Memory-Strukturen, die als System Global Area (SGA) bezeichnet werden, sowie aus Hintergrundprozessen. Die Kombination aus SGA und Oracle-Hintergrundprozessen wird als Oracle-Instanz bezeichnet. Oracle-Instanzen sind immer mit einer einzelnen Datenbank verknüpft. Unter bestimmten Umständen können aber auch Oracle-Datenbanken mehrere Instanzen besitzen.

SQL Server-Instanz

Jede SQL Server-Instanz besteht aus einem Satz von Diensten mit bestimmten Einstellungen für Sortierungen und andere Optionen[8]. Bei der Instanz kann es sich um eine Standardinstanz oder um eine benannte Instanz handeln. SQL Server-Instanzen können mit vielen Datenbanken verknüpft sein, wobei jede Instanz unabhängig von den anderen Instanzen ausgeführt wird. Zwar unterstützt der SQL Server 2005 mehrere Instanzen von SQL Server auf einem einzelnen Server oder Prozessor aber die Standardinstanz kann nur eine Instanz sein, alle anderen Instanzen müssen benannte Instanzen sein.

[7] Vgl. [Best05] Kapitel 1, Seite 13
[8] Vgl. [MSOD3]

3 Datenbankobjekte

3.1 Datentypzuordnungen

„Oracle-Datentypen und SQL Server-Datentypen stimmen nicht immer exakt überein. Für Fälle, in denen eine einzelne Datentypzuordnung unklar ist, gibt es noch alternative Datentypzuordnungen“[9]. Doch vorerst sollte überprüft werden, ob der SQL Server signifikantere Datentypen anbietet, damit nicht nachteilige Datentypen weiter verwendet werden. Weiterhin ist darauf zu achten, dass die Wertebereiche entweder identisch oder größer als die in Oracle verwendeten Datentypen sein müssen.
Die folgende Tabelle zeigt die standardmäßige Zuordnung von Datentypen von Oracle zu SQL Server. Die Spalte "Alternativen" gibt an, ob alternative Zuordnungen verfügbar sind:

Datentypzuordnungen von Oracle zu SQL Server

Oracle-Datentyp	SQL Server-Datentyp	Alternativen
BFILE	VARBINARY(MAX)	Ja
BLOB	VARBINARY(MAX)	Ja
CHAR([1-2000])	CHAR([1-2000])	Ja
CLOB	VARCHAR(MAX)	Ja
DATE	DATETIME	Ja
FLOAT	FLOAT	Nein
FLOAT([1-53])	FLOAT([1-53])	Nein
FLOAT([54-126])	FLOAT	Nein
INT	NUMERIC(38)	Ja
INTERVAL	DATETIME	Ja
LONG	VARCHAR(MAX)	Ja
LONG RAW	IMAGE	Ja
NCHAR([1-1000])	NCHAR([1-1000])	Nein
NCLOB	NVARCHAR(MAX)	Ja

[9] [MSDN2]

LONG RAW	IMAGE	Ja
NCHAR([1-1000])	NCHAR([1-1000])	Nein
NCLOB	NVARCHAR(MAX)	Ja
NUMBER	FLOAT	Ja
NUMBER([1-38])	NUMERIC([1-38])	Nein
NUMBER([0-38],[1-38])	NUMERIC([0-38],[1-38])	Ja
NVARCHAR2([1-2000])	NVARCHAR([1-2000])	Nein
RAW([1-2000])	VARBINARY([1-2000])	Nein
REAL	FLOAT	Nein
ROWID	CHAR(18)	Nein
TIMESTAMP	DATETIME	Ja
UROWID	CHAR(18)	Nein
VARCHAR2([1-4000])	VARCHAR([1-4000])	Ja

Tabelle 2: Datentypzuordnungen von Oracle zu SQL Server

Die folgende Tabelle zeigt die standardmäßige Zuordnung von Datentypen von SQL Server zu Oracle:

Datentypzuordnungen von SQL Server zu Oracle

SQL Server-Datentyp	Oracle-Datentyp
BIGINT	NUMBER(19,0)
BINARY(1-2000)	RAW(1-2000)
BINARY(2001-8000)	BLOB
BIT	NUMBER(1)
CHAR(1-2000)	CHAR(1-2000)
CHAR(2001-4000)	VARCHAR2(2001-4000)
CHAR(4001-8000)	CLOB
DATETIME	DATE
DECIMAL(1-38, 0-38)	NUMBER(1-38, 0-38)
DOUBLE PRECISION	FLOAT
FLOAT	FLOAT
IMAGE	BLOB
INT	NUMBER(10,0)

MONEY	NUMBER(19,4)
NCHAR(1-1000)	CHAR(1-1000)
NCHAR(1001-4000)	NCLOB
NTEXT	NCLOB
NUMERIC(1-38, 0-38)	NUMBER(1-38, 0-38)
NVARCHAR(1-1000)	VARCHAR2(1-2000)
NVARCHAR(1001-4000)	NCLOB
NVARCHAR(MAX)	NCLOB
REAL	REAL
SMALLDATETIME	DATE
SMALLINT	NUMBER(5,0)
SMALLMONEY	NUMBER(10,4)
SQL_VARIANT	<nicht unterstützt>
SYSNAME	VARCHAR2(128)
TEXT	CLOB
TIMESTAMP	RAW(8)
TINYINT	NUMBER(3,0)
UNIQUEIDENTIFIER	CHAR(38)
VARBINARY(1-2000)	RAW(1-2000)
VARBINARY(2001-8000)	BLOB
VARCHAR(1-4000)	VARCHAR2(1-4000)
VARCHAR(4001-8000)	CLOB
VARBINARY(MAX)	BLOB
VARCHAR(MAX)	CLOB
XML	NCLOB

Tabelle 3: Datentypzuordnungen von SQL Server zu Oracle

Datum

Während in SQL Server die Datentypen DATETIME und SMALLDATETIME für Datums- und Tageszeitangaben benutzt werden, stellt Oracle die Datentypen TIMESTAMP und DATE zur Verfügung[10]:

[10] [MurDOC]

Datums- und Zeitangaben in SQL Server

Datentyp	Bereich	Genauigkeit
DATETIME	Zwischen dem 1. Januar 1753 und dem 31. Dezember 9999.	3,33 Millisekunden
SMALL-DATETIME	Zwischen dem 1. Januar 1900 und dem 6. Juni 2079.	1 Minute

Tabelle 4: Datums- und Zeitangaben in SQL Server

Datums- und Zeitangaben in Oracle

Datentyp	Bereich	Genauigkeit
TIME-STAMP	Zwischen dem 1. Januar 4712 v. Chr. und dem 31. Dezember 9999 n. Chr.	Ein einhundertmillionstel (1/100000000) einer Sekunde
DATE	Zwischen dem 1. Januar 4712 v. Chr. und dem 31. Dezember 9999 n. Chr.	1 Sekunde

Tabelle 5: Datums- und Zeitangaben in Oracle

Datumskonvertierungen sollten manuell erstellt werden, weil Konvertierungsprogramme nicht die Logik hinter einer Tabelle verstehen können. Wenn beispielsweise eine Spalte Geburtstag mit dem Oracle-Datentyp DATE einer Tabelle Personen in eine Spalte mit dem SQL Server-Datentyp DATETIME konvertiert wird, dann werden diese Spalten noch mit überflüssigen Zeitangaben aufgeführt (Siehe Abschnitt 5.3).

FLOAT und NUMBER

Die Anzahl der Dezimalstellen (scale) und die Genauigkeit (precision), die während der Zuordnung der Datentypen FLOAT und NUMBER angegeben werden, sind von den Parametern abhängig, die in der Spalte der Oracle-Datenbank angegeben wurde, die diese Datentypen verwendet. Die Zahl 123,45 hat z. B. eine Genauigkeit von 5 (Ziffern) und 2 Dezimalstellen.

„Bei Oracle können Zahlen mit Werten für 'scale' größer als 'precision' definiert werden, z. B. NUMBER(4,5). In SQL Server muss jedoch 'precision' größer oder gleich 'scale' sein. Um sicherzustellen, dass keine Daten abgeschnitten werden, wenn bei Oracle 'scale' größer ist als 'precision', sollte

'precision' bei der Zuordnung auf denselben Wert festgelegt werden wie 'scale': NUMBER(4,5) wird also beispielsweise NUMERIC(5,5) zugeordnet“[11].

LOB-Typen (Large Object)

„Oracle unterstützt bis zu 4 Gigabyte (GB), SQL Server bis zu 2 GB“[12]. Daten aus Oracle, die über 2 GB sind, müssen deshalb abgeschnitten werden.

Indizes

Indizes müssen neu generiert werden, wobei geprüft werden sollte, welche Optimierungen in SQL Server möglich sind.

NULL-Verhalten

Unique

Für Spalten, die von Unique-Einschränkungen betroffen sind, muss ein Wert ungleich NULL angegeben sein. Oracle und SQL Server verarbeiten NULL-Werte unterschiedlich: Oracle lässt mehrere Zeilen mit NULL-Werten für Spalten zu, bei denen NULL-Werte zulässig sind und die in UNIQUE-Einschränkungen oder -Indizes eingeschlossen sind. SQL Server erzwingt die Eindeutigkeit, indem nur eine einzige Zeile mit einem NULL-Wert für eine Spalte zulässig ist.

ORDER BY-Klauseln

Auch hier sollten einige Punkte über die unterschiedliche Behandlung von NULL-Werten in Oracle und SQL Server beachtet werden:

- In SQL Server sind NULL-Werte die niedrigsten Werte in einer geordneten Liste. In einer aufsteigenden Liste stehen NULL-Werte an erster Stelle
- In Oracle sind NULL-Werte die größten Werte in einer geordneten Liste. Als default stehen NULL-Werte in einer aufsteigenden Liste an erster Stelle.

[11] [12] [MSDN2]

ISNULL in String-Konkatenation

Oracle und SQL Server geben unterschiedliche Ergebnisse zurück wenn in String-Konkatenationen NULL-Werte enthalten sind. Oracle behandelt die NULL-Werte wie einen leeren Zeichensatz, während der SQL Server NULL zurück gibt. Die folgende Tabelle zeigt, dass in Oracle NULL niemals äquivalent ist mit Null:

NULL-Konstrukte in Oracle und SQL Server

NULL-Konstrukt	SQL Server	Oracle
where col1 = NULL	datenabhängig	FALSE
where col1 != NULL	datenabhängig	TRUE
where col1 IS NULL	datenabhängig	datenabhängig
where col1 IS NOT NULL	datenabhängig	datenabhängig
where NULL = NULL	TRUE	FALSE

Tabelle 6: NULL-Konstrukte in Oracle und SQL Server

Falls solche SQL Server-Abfragen vorliegen

```
WHERE col1 = NULL
```

sollten diese in solche Oracle-Abfragen umgeschrieben werden

```
WHERE col1 IS NULL
```

3.2 Unterschiede zwischen ANSI-SQL und ORACLE-SQL

Syntaxunterschiede

Oracle Database 10g unterstützt den ANSI SQL:2003-Standard, während Microsoft SQL Server 2005 den ANSI SQL:1999-Standard unterstützt. Bei Oracle wird aber dennoch in der Praxis meistens die ORACLE-Syntax verwendet, die sich in entscheidenden Punkten von dem ANSI-Standard unterscheidet.

Die folgende Tabelle zeigt die in Oracle und SQL Server verwendete unterschiedliche Syntax für dieselbe SQL-Anweisung:

SQL-Syntaxunterschiede

Beschreibung	Oracle	SQL Server

Left Outer Join	*Bis Oracle 9i nur:*
	`WHERE col1 = col2(+)`
	Ab Oracle 9i:
	`FROM tab1 LEFT OUTER JOIN tab2 ON tab1.col1 = tab2.col2`
		`FROM tab1 LEFT OUTER JOIN tab2 ON tab1.col1 = tab2.col2`

Right Outer Join	*Bis Oracle 9i nur:*
	`WHERE col1(+) = col2`
	Ab Oracle 9i:
	`FROM tab1 RIGHT OUTER JOIN tab2 ON tab1.col1 = tab2.col2`
		`FROM tab1 RIGHT OUTER JOIN tab2 ON tab1.col1 = tab2.col2`

Full Outer Join	*Ab Oracle 9i:* `FROM tab1 FULL OUTER JOIN tab2 ON tab1.col1 = tab2.col2`	`FROM tab1 FULL OUTER JOIN tab2 ON tab1.col1 = tab2.col2`

SELECT ohne FROM	`SELECT 'hello world' FROM DUAL;`
	(Pseudo-Tabelle DUAL nötig)
		`SELECT 'hello world'`
		oder
		`SELECT 3*6`

INSERT	`INSERT INTO [user.]{table | view}[@dblink][(column [, column]...)]{VALUES (expression [, expression]...) | query...};` *<INTO ist obligatorisch>*	`INSERT [INTO] [[database.]owner.] {table | view}[(column [, column]...)]{VALUES (expression [,expression]...) | query}` *<INTO ist optional>*
Transformation großer Datenmengen	`CREATE TABLE ... AS SELECT ...`	`SELECT ... INTO`
Subtraktion zweier SELECTS	`SELECT ... MINUS SELECT ...`	`SELECT ... EXCEPT SELECT ...`

SELECT INTO-Anweisung	INSERT INTO tab1 SELECT col1,col2,col3 FROM tab2 WHERE ...	SELECT col1, col2, col3 INTO tab1 FROM tab2 WHERE ...
Spalte löschen	*Ab Oracle 8i:* ALTER TABLE table_name DROP COLUMN column_name UNUSE …	ALTER TABLE table_name DROP COLUMN column_name
Verschachtelte Tabellen (nested tables)	*Ab Oracle 9i:* NESTED TABLE tab1 store AS tab2	*<nicht unterstützt>*
Readonly VIEW	CREATE VIEW ... **WITH READONLY**	GRANT SELECT ...
Sequence	CREATE SEQUENCE seq_name ...	IDENTITY (s, i) *S = Anfangswert* *i = inkrementeller Wert* *Siehe Abschnitt 5.4*
Synonym	CREATE SYNONYM s_name FOR obj_name;	*Lösungsmöglichkeit:* *Siehe Abschnitt 5.4*
Sicherungspunkt festlegen	SAVEPOINT	SAVE TRANSACTION
Lockmodi	SHARE	TABLOCK, HOLDLOCK
	EXCLUSIVE	TABLOCKX, HOLDLOCK
	ROW SHARE SHARE UPDATE = ROW SHARE	ROWLOCK, HOLDLOCK
	ROW EXCLUSIVE	ROWLOCK, XLOCK, HOLDLOCK
	SHARE ROW EXCLUSIVE	TABLOCK, XLOCK, HOLDLOCK
	ROW-Level	*<keine Entsprechung>*
Lockmode-Optionen	PARTITION, SUBPARTITION, @dblink, NOWAIT	*<nicht unterstützt>*
Zeitspanne zur Aufhebung einer Sperre festlegen	*Keine Zeitspannenangabe möglich, außer* **NOWAIT** *agiert wie* **"LOCK_TIMEOUT 0"**	SET LOCK_TIMEOUT timeout_period
Reservierung des Indexbereichs	PCTFREE=0	FILLFACTOR=100

DESCRIBE	`DESCRIBE table_name` *(SQLPlus-Befehl)*	`sp_help oder sp_columns`
Pseudospalte ROWID	`SELECT ROWID, SUBSTR(ROWID, 1, 6) object, ...`	*<keine Entsprechung>* *Lösungsmöglichkeit:* *Siehe Abschnitt 5.4*
Pseudospalte ROWNUM	`SELECT ROWNUM, empno, ename FROM scott.emp e ...`	*<keine Entsprechung>* *Lösungsmöglichkeit:* *Siehe Abschnitt 5.4*
Spalten-Alias	`SELECT col1 employees FROM table`	`SELECT employees=col1 FROM table`
Alias	`SELECT name alias_name`	
	`SELECT name AS alias_name`	

Tabelle 7: SQL-Syntaxunterschiede[13]

Syntaxunterschiede der prozeduralen Spracherweiterungen

PL/SQL und T-SQL sind rein herstellerbezogene prozedurale Erweiterungen des SQL-Standards. Sie bieten zwar viele derselben Konstrukte aber haben jedoch – außer die Grundkonzeption und Funktionalität betreffend – hinsichtlich der Sprachsyntax nicht allzu viele Gemeinsamkeiten.

Die folgende Tabelle zeigt einige Syntaxunterschiede von PL/SQL und T-SQL:

Syntaxunterschiede von PL/SQL und T-SQL

Beschreibung	Oracle PL/SQL	SQL Server T-SQL
Lokale Variable in DECLARE initialisieren	`DECLARE var_name type := value;`	*<nicht unterstützt>*
Konstante deklarieren	`DECLARE var_name CONSTANT type := value;`	*<nicht unterstützt>*
Variable zuweisen	`Var_name := value` `SELECT value INTO var_name`	`SET @var_name = value` `SELECT @var_name = value`
%TYPE	`DECLARE guthaben NUMBER(7,2); schuld guthaben%TYPE;`	*<nicht unterstützt>*

[13] Vgl. [MSTN1]

Collection	TYPE type_name IS TABLE OF element_type [NOT NULL];	*<nicht unterstützt>* *Lösungsmöglichkeit:* *Siehe Abschnitt 5.3*
Cursor deklarieren	CURSOR cur_name *(params)* IS *SELECT*;	DECLARE cur_name CURSOR (params) FOR SELECT
Cursor in eine Variable zuweisen	FETCH cur_name INTO var_name	FETCH (params) FROM cur_name INTO @var_name
Cursor öffnen	OPEN cur_name (params);	OPEN cur_name
Cursorverweis entfernen	*<nicht erforderlich>*	DEALLOCATE cur_name
If-Anweisung	IF ... THEN, ELSIF ... THEN, ELSE, ENDIF; *Alternativ:* CASE WHEN ... THEN...; ELSE ... ; END CASE;	IF ... [BEGIN ... END] ELSE ... [BEGIN ... END] ELSE IF ... CASE ...
While-Schleife	WHILE ... LOOP END LOOP;	WHILE ... [BEGIN ... END]
Schleifen-kontrolle	FOR ... END LOOP; LOOP ... END LOOP;	*<nicht unterstützt>*
Schleifen-abbruch	EXIT, EXIT WHEN	BREAK
While-Schleifen-Neustart	*<nicht unterstützt>*	CONTINUE
Print output	*Realisiert über die Funktion:* DBMS_OUTPUT.PUT_LINE	PRINT
Raise error	RAISE_APPLICATION_ERROR	RAISERROR
Pakete	CREATE OR REPLACE PACKAGE package_Name	*<nicht unterstützt>* *Siehe Abschnitt 5.4*
Trigger	BEFORE TRIGGER	INSTEAD OF TRIGGER
	AFTER TRIGGER	AFTER TRIGGER
	INSTEAD OF TRIGGER	INSTEAD OF TRIGGER
	ROW LEVEL TRIGGER	*imitiert durch* CURSOR
Abschlusszeichen	Semikolon (;)	*<nicht erforderlich>*

Tabelle 8: Syntaxunterschiede von PL/SQL und T-SQL

Über weitere Möglichkeiten für Syntaxzuordnungen und Lösungsmöglichkeiten für einige in SQL Server nicht unterstützte Syntaxzuordnungen sei nochmals auf Abschnitt 5.4 verwiesen.

Operatoren

Die meisten Operatoren in Oracle und SQL Server stimmen überein. Hier nur ein Operator, der sich unterscheidet[14]:

Operatorunterschiede

Beschreibung	Oracle	SQL Server
Konkatenation	`string1 || string2`	`string1 + string2`

Tabelle 9: Operatorunterschiede

3.3 Namenskollisionen

Reservierte Schlüsselwörter

Beide Datenbankmanagementsysteme verwenden eine Fülle von reservierten Schlüsselwörtern zum Definieren, Bearbeiten und Zugreifen auf Datenbanken[15]. Keine dieser reservierten Schlüsselwörter sollten für Objektnamen (Tabellen, Indizes, Views, …) und Objektbezeichner verwendet werden. Doch einige Objekte, die ursprünglich in Oracle erstellt wurden, könnten Benennungen aufweisen, die für reservierte Schlüsselwörter in SQL Server verwendet werden. Damit keine Namenskollisionen auftreten, sollten diese Objektnamen und Objektbezeichner vor einer Migration umbenannt werden.

Die folgende Tabelle führt reservierte Schlüsselwörter von SQL Server auf, die aber in Oracle als Objektnamen und –bezeichner verwendet werden dürfen, da sie in Oracle nicht als reservierte Schlüsselwörter deklariert wurden[16]:

[14] Vgl. [MSTN1]
[15] Vgl. [MSDN3]
[16] Vergleiche hierzu alle reservierten Schlüsselwörter in der Oracle-Onlinedokumentation unter [OraDOC1] und [OraDOC2]

Reservierte Schlüsselwörter des SQL Server

BROWSE	IDENTITY	READTEXT
CLUSTERED	IDENTITY_INSERT	RECONFIGURE
COLLATE	IDENTITYCOL	REPLICATION
COMPUTE	INNER	RESTORE
CONTAINS	KILL	RESTRICT
CONTAINSTABLE	LEFT	REVERT
CROSS	LINENO	ROWCOUNT
CURRENT_DATE	LOAD	ROWGUIDCOL
CURRENT_TIME	NOCHECK	RULE
CURRENT_TIMESTAMP	NONCLUSTERED	SESSION_USER
CURRENT_USER	OFFSETS	SETUSER
DBCC	OPENDATASOURCE	SYSTEM_USER
DEALLOCATE	OPENQUERY	TABLESAMPLE
DENY	OPENROWSET	TEXTSIZE
DISK	OPENXML	TOP
DISTRIBUTED	OUTER	TRAN
DUMMYRECONFIGURE	OVER	TSEQUAL
FILLFACTOR	PERCENT	UNPIVOT
FREETEXT	PIVOT	UPDATETEXT
FREETEXTTABLE	PRINT	WAITFOR
FULL	PROC	WRITETEXT
HOLDLOCK	RAISERROR	

Tabelle 10: Reservierte Schlüsselwörter des SQL Server

Reservierte ODBC-Schlüsselwörter

Die folgenden Wörter sind für die Verwendung in ODBC-Funktionsaufrufen reserviert. Damit in SQL Server die Kompatibilität mit Treibern sichergestellt ist, die die zentrale (core) SQL-Grammatik unterstützen, sollten diese Schlüsselwörter nicht in dem Quellsystem als Objektnamen und –bezeichner auftauchen[17].

Diese Tabelle listet die reservierten ODBC-Schlüsselwörter des SQL Server auf, die in Oracle nicht als reservierte Schlüsselwörter auftreten:

Reservierte ODBC-Schlüsselwörter des SQL Server

ABSOLUTE	DIAGNOSTICS	PAD
ACTION	DISCONNECT	PARTIAL
ADA	DOMAIN	POSITION
ARE	END-EXEC	PREPARE

[17] [MSDN3]

ASSERTION	EXTRACT	PRESERVE
BIT	FIRST	RELATIVE
BIT_LENGTH	GET	RIGHT
CASCADED	GLOBAL	SCROLL
CAST	INPUT	SQLCA
CATALOG	INSENSITIVE	SQLWARNING
CHAR_LENGTH	LAST	SUBSTRING
CHARACTER_LENGTH	LOWER	TRANSLATE
COLLATION	MATCH	TRANSLATION
CONNECTION	NAMES	TRIM
CORRESPONDING	NO	UNKNOWN
DEFERRABLE	OCTET_LENGTH	UPPER
DESCRIPTOR	OUTPUT	USAGE

Tabelle 11: Reservierte ODBC-Schlüsselwörter des SQL Server

Zukünftige Schlüsselwörter

Die in dieser Tabelle aufgelisteten Schlüsselwörter werden möglicherweise in zukünftigen SQL Server-Versionen reserviert, wenn neue Features implementiert werden[18]. Aus diesem Grund sollte auch bei diesen Wörtern darauf geachtet werden, dass sie in dem Quellsystem nicht benutzt wurden:

Zukünftige Schlüsselwörter des SQL Server

BREADTH	INITIALIZE	RETURNS
CLASS	INOUT	ROLLUP
COMPLETION	ITERATE	ROUTINE
CUBE	LATERAL	SCOPE
CURRENT_PATH	LESS	SEARCH
CURRENT_ROLE	LOCALTIME	SETS
DEPTH	LOCALTIMESTAMP	SPECIFIC
DEREF	LOCATOR	SPECIFICTYPE
DESCRIBE	MODIFIES	SQLEXCEPTION
DESTROY	OPERATION	STATE
DESTRUCTOR	ORDINALITY	STRUCTURE
DYNAMIC	PATH	THAN
EVERY	POSTFIX	TREAT
FULLTEXTTABLE	PREFIX	UNNEST
GROUPING	PREORDER	WITHOUT
HOST	READS	
IGNORE	RECURSIVE	

Tabelle 12: Zukünftige Schlüsselwörter des SQL Server

[18] [MSDN3]

3.4 Maximal zulässige Größen der SQL Server Datenbankmodul- Objekte

Die maximal zulässige Größe für einige Objekte von Oracle und SQL Server unterscheiden sich[19]. Die Größe der Objekte, die migriert werden sollen, muss mit der maximal zulässigen Größe der entsprechenden Objekte in SQL Server übereinstimmen.

Die Tabelle Maximale Kapazität für SQL Server 2005[20] in dem Anhang A gibt die maximale Größe und Anzahl verschiedener in SQL Server 2005-Datenbanken definierter oder in T-SQL Anweisungen referenzierter Objekte an.

[19] Vgl. [MSDN4]
[20] Vgl. [MSDN5]

4 Sperrkonzepte

Obwohl Sperren in beiden Datenbankmanagementsystemen automatisch und ohne explizite Anweisungen des Benutzers ausgeführt werden ist es unter Umständen aber trotzdem notwendig, die unterschiedlichen Sperrkonzepte zu analysieren. Durch veränderte Sperren kann es nämlich bei konkurrierenden Zugriffen zu Deadlocks oder starken Performance-Einbußen kommen, die mit den ursprünglichen Sperren nicht vorgekommen wären.

4.1 LOCKING

Durch einen „LOCK“ der Tabellen wird eine serielle Abarbeitung von Transaktionen erzwungen, indem bei Datenänderungen die beteiligten Tabellen für andere Benutzer gesperrt werden[21].

4.1.1 LOCKING in SQL Server

Das LOCKING in SQL Server geschieht völlig automatisch und erfordert keine manuellen Eingriffe.
Die folgende Tabelle zeigt die Ressourcen-Sperrmodi, die das Datenbankmodul verwendet[22]:

Automatische Sperrmodi des SQL Server

Sperrmodus	Beschreibung
S (Shared)	Wird für Lesevorgänge verwendet, die Daten nicht ändern oder aktualisieren, wie z. B. SELECT-Anweisungen.
U (Update)	Wird für Ressourcen verwendet, die aktualisiert werden können. Verhindert eine gängige Form des Deadlocks, die auftritt, wenn mehrere Sitzungen Ressourcen lesen, sperren und anschließend möglicherweise aktualisieren.

[21] Vgl. [FaeskScript]
[22] [MSOD4]

X (Exclusive)	Wird bei Datenänderungen wie INSERT-, UPDATE- oder DELETE-Vorgängen verwendet. Stellt sicher, dass nicht mehrere Aktualisierungen an derselben Ressource gleichzeitig vorgenommen werden können.
Intent	Wird verwendet, um eine Sperrhierarchie unter den "Beabsichtigten Sperren" zu erstellen. *(Siehe Tabelle Beabsichtigte Sperren)*
Schema	Schemaänderungssperren (Sch-M) werden verwendet, wenn für eine Tabelle ein DDL-Vorgang ausgeführt wird, wie etwa das Hinzufügen einer Spalte oder Löschen einer Tabelle.
BU (Bulk Update)	Wird beim Massenkopieren von Daten in eine Tabelle verwendet, wenn der TABLOCK-Hinweis angegeben ist. Massenaktualisierungssperren (BU) werden verwendet, damit mehrere Threads gleichzeitig Daten in dieselbe Tabelle laden können, während sie zugleich anderen Prozessen, die keine Daten massenkopieren, keinen Zugriff auf die Tabelle gewähren.
Schlüssel-bereich	Schützt den von einer Abfrage gelesenen Zeilenbereich, wenn die serialisierbare Transaktionsisolationsstufe verwendet wird. Stellt sicher, dass keine anderen Transaktionen Zeilen einfügen können, die von den Abfragen der serialisierbaren Transaktion berücksichtigt werden könnten, falls diese erneut ausgeführt würden.

Tabelle 13: Automatische Sperrmodi des SQL Server

Hinweis

Wenn der SQL Server für Lesevorgänge einen SHARED LOCK auf eine Tabelle hält, kann keine andere Transaktion einen EXCLUSIVE LOCK auf die betroffene Tabelle absetzen[23]. Alle Anweisungen, die die Daten aktualisieren möchten, benötigen aber einen EXCLUSIVE LOCK. Das bedeutet, dass Lesevorgänge in SQL Server Schreibvorgänge auf denselben Datensatz solange blockieren bis die Transaktion, die die SELECT-Anweisung ausführt,

[23] Vgl. [MurDOC]

ein COMMIT oder ein ROLLBACK absetzt. Das führt dazu, dass zwei Transaktionen in einer seriellen Reihenfolge ausgeführt werden, wann immer eine Transaktion T1 die Daten liest und eine andere Transaktion T2 die Daten ändern und danach wieder lesen möchte.
In Oracle jedoch, werden Schreibvorgänge nicht durch Lesevorgänge auf demselben Datensatz blockiert, da die Daten vor einem Tabellen-UPDATE in den UNDO-Segmenten gespeichert werden. Aus diesem Grund können in Oracle Transaktionen parallel ausgeführt werden.

Falls logische SQL Server-Transaktionen automatisch in logische Oracle-Transaktionen konvertiert werden, können die vorher seriell ausgeführten Transaktionen in Oracle einen Deadlock verursachen.

Beabsichtigte Sperren

Das Datenbankmodul verwendet beabsichtigte Sperren, um das Platzieren einer gemeinsamen (S) oder exklusiven Sperre (X) auf eine Ressource zu schützen, die sich weiter unten in der Sperrhierarchie befindet. Die Bezeichnung 'beabsichtigte Sperre' bedeutet, dass beabsichtigte Sperren vor Sperren auf untergeordneten Ebenen eingerichtet werden, und damit die Absicht ausdrücken, Sperren auf untergeordneten Ebenen zu platzieren.
Beabsichtigte Sperren werden aus zwei Gründen verwendet:

- Um zu verhindern, dass andere Transaktionen Ressourcen übergeordneter Ebenen ändern und damit die Sperren untergeordneter Ebenen ungültig werden.
- Um die Effizienz des Datenbankmoduls beim Erkennen von Sperrkonflikten auf einer höheren Granularitätsebene zu steigern.

Die folgende Tabelle gibt eine Übersicht über die in SQL Server verwendeten beabsichtigen Sperren:

Beabsichtigte Sperren des SQL Server

Sperrmodus	Beschreibung
Beabsichtigte gemeinsame Sperre (IS)	Schützt angeforderte oder eingerichtete gemeinsame Sperren bestimmter (aber nicht aller) Ressourcen untergeordneter Ebenen in der Hierarchie.
Beabsichtigte exklusive Sperre (IX)	Schützt angeforderte oder eingerichtete exklusive Sperren bestimmter (aber nicht aller) Ressourcen untergeordneter Ebenen in der Hierarchie. IX ist eine Obermenge von IS und schützt auch vor Anforderung gemeinsamer Sperren auf Ressourcen untergeordneter Ebenen in der Hierarchie.
Gemeinsame Sperre mit beabsichtigter exklusiver Sperre (SIX)	Schützt angeforderte oder eingerichtete gemeinsame Sperren aller Ressourcen untergeordneter Ebenen in der Hierarchie sowie beabsichtigte exklusive Sperren bestimmter (aber nicht aller) Ressourcen untergeordneter Ebenen in der Hierarchie.
Beabsichtigte Aktualisierungs-sperre (Intent Update, IU)	Schützt angeforderte oder eingerichtete Aktualisierungssperren aller Ressourcen untergeordneter Ebenen in der Hierarchie.
Gemeinsame beabsichtigte Aktualisierungs-sperre (Shared Intent Update, SIU)	Eine Kombination der Sperren vom Typ S und IU, die sich aus der separaten Einrichtung dieser Sperren und dem gleichzeitigen Beibehalten beider Sperren ergibt. Falls eine Transaktion eine Abfrage mit dem PAGLOCK-Hinweis und anschließend einen Aktualisierungsvorgang ausführt. Die Abfrage mit dem PAGLOCK-Hinweis richtet die S-Sperre ein, wohingegen der Aktualisierungsvorgang die IU-Sperre einrichtet.
Exklusive beabsichtigte Aktualisierungs-sperre (Update intent exclusive, UIX)	Eine Kombination der Sperren vom Typ U und IX, die sich aus dem separaten Einrichten dieser Sperren und dem gleichzeitigen Beibehalten beider Sperren ergibt.

Tabelle 14: Beabsichtigte Sperren des SQL Server

4.1.2 LOCKING in Oracle

Das LOCKING in Oracle geschieht völlig automatisch und erfordert keine manuellen Eingriffe. “Bei lesenden Zugriffen werden keine Sperren aufgebaut. Bei schreibenden Zugriffen werden auf einzelne Zeilen (DML) oder der kompletten Tabelle (DDL) EXCLUSIVE LOCKS bzw. ROW EXCLUSIV LOCKS gesetzt.
In Oracle wird je nach SQL-Befehl ohne weitere Voreinstellung ein bestimmter Sperrmodus eingestellt“[24]:

Automatische Sperrmodi von Oracle

SQL-Anweisung	Sperrmodus
`SELECT ... FROM table`	keiner
`INSERT INTO table ...`	ROW EXCLUSIV
`UPDATE table ...`	ROW EXCLUSIV
`DELETE FROM table ...`	ROW EXCLUSIV
`ALTER TABLE ...`	EXCLUSIV

Tabelle 15: Automatische Sperrmodi von Oracle

Zusätzlich können manuell Sperren gesetzt werden, die aber automatisch nach Beendigung der Transaktion wieder freigegeben werden. Dabei gilt zu beachten, dass jede DDL-Anweisung ein automatisches COMMIT enthält. In der folgenden Tabelle werden die in Oracle möglichen Sperrmodi dargestellt, mit denen manuelle Sperren auf Tabellen möglich sind:

Manuelle Sperrmodi von Oracle

Sperrmodus	Auswirkung
SHARE	Erlaubt lesenden, aber keinen schreibenden Zugriff.
EXCLUSIV	Sperrt die Tabelle für alle Zugriffe.
ROW SHARE	Auf der betroffenen Tabelle kann kein EXCLUSIVE LOCK ausgelöst werden.

[24] [FaeskScript]

ROW EXCLUSIVE	Sperrt einzelne Zeilen und verhindert EXCLUSIV LOCKS und SHARED LOCKS auf den Tabellen.
SHARE ROW EXCLUSIV	Verhindert EXCLUSIVE LOCKS, SHARED LOCKS und UPDATE von Daten.

Tabelle 16: Manuelle Sperrmodi von Oracle

4.2 Lesekonsistenz

Bei einem Mehrbenutzerbetrieb kann das Problem der Lesenkonsistenz auftreten. „Bei länger andauernden Anfragen bzw. Transaktionen stellt sich die Frage, auf welchem Datenbankzustand wird die Anfrage eigentlich ausgewertet, wenn parallel andere Benutzer die zu lesenden Daten ändern“[25].

4.2.1 Lesekonsistenz in SQL Server

Um sicherzustellen, dass die Daten aller Anfragen einer Transaktion vom gleichen Zeitpunkt (Start der Transaktion) stammen stellt der SQL Server die HOLDLOCK-Funktion zur Verfügung. Ein HOLDLOCK auf einem Lesevorgang führt gleichzeitig noch ein SHARED LOCK mit sich. Mit HOLDLOCK können mehrere Benutzer gleichzeitig Lesevorgänge starten, ohne sich gegenseitig zu blockieren.
Wenn aber einer der Benutzer ein UPDATE auf den gelesenen Datensätzen ausführen möchte, blockiert HOLDLOCK dieses UPDATE solange, bis die anderen Benutzer ein COMMIT oder ein ROLLBACK absetzen oder selber einen UPDATE-Versuch unternehmen und ein Deadlock tritt auf. Das bedeutet, solange noch die aktuelle Transaktion ausgeführt wird, verhindert HOLDLOCK andere Transaktionen daran, UPDATE-Vorgänge auf demselben Datensatz auszuführen.

[25] [FaeskScript]

4.2.2 Lesekonsistenz in Oracle

Oracle stellt automatisch sicher, dass eine Lesekonsistenz in jedem Fall gewährleistet werden kann. Das bedeutet:

- Es wird sichergestellt, dass die ausgelesenen Datensätze zu diesem Zeitpunkt konsistent sind und sich während des Lesevorgangs nicht ändern.
- Es wird sichergestellt, dass Lesevorgänge nicht auf andere Lese- oder Schreibvorgänge auf den gleichen Daten warten müssen.
- Es wird sichergestellt, dass Schreibvorgänge nicht auf andere Lesevorgänge auf den gleichen Daten warten müssen.
- Es wird sichergestellt, dass Schreibvorgänge auf andere Schreibvorgänge nur warten, wenn sie versuchen identische Zeilen in nebenläufigen Transaktionen zu ändern.

Für die Lesekonsistenz werden hierfür die UNDO-Segmente benötigt. Dabei gilt:

1. Geänderte Datensätze von offenen (nicht durch ROLLBACK oder COMMIT) abgeschlossenen Transaktionen werden als solche gekennzeichnet. Ein Lesezugriff auf diese Sätze wird automatisch auf einen gültigen Eintrag in den UNDO-Segmenten umgeleitet.
2. Geänderte Datensätze einer geschlossenen Transaktion werden mit einem Zeitstempel (SCN) versehen, so dass ein Lesezugriff über einen Zeitvergleich ermitteln kann, ob dieser Datensatz zum Zeitpunkt des Starts gültig war. Ansonsten wird wiederum ein gültiger Datensatz in den UNDO-Segmenten gesucht.

4.3 Isolationsstufen

“Transaktionen geben eine Isolationsstufe an, mit der definiert wird, bis zu welchem Ausmaß eine Transaktion von Ressourcen- oder Datenänderungen isoliert sein muss, die von anderen Transaktionen durchgeführt werden[26]“. Wenn das Ändern und Lesen von Daten durch mehrere Benutzer nicht verwaltet wird, können Probleme mit der Parallelität auftreten. Wenn beispielsweise mehrere Benutzer zeitgleich auf eine Datenbank zugreifen, kann es vorkommen, dass die Transaktionen der Benutzer gleichzeitig für dieselben Daten Vorgänge ausführen. Im Folgenden werden einige Parallelitätsprobleme aufgeführt:

- **Dirty Read:** Dieses Problem tritt dann auf, wenn eine zweite Transaktion eine Zeile auswählt, die von einer anderen Transaktion aktualisiert wird. Die zweite Transaktion liest Daten, für die noch kein COMMIT ausgeführt wurde und die von der Transaktion, die die Zeile aktualisiert, noch geändert werden können.
- **Non Repeatable Read:** Ein Non Repeatable Read tritt dann ein, wenn eine zweite Transaktion mehrmals auf dieselbe Zeile zugreift und jedes Mal verschiedene Daten liest. Für dieselbe Zeile werden mehrere Lesevorgänge durchgeführt, wobei jedes Mal die Informationen von einer anderen Transaktion geändert werden.
- **Phantom Read:** Ein Phantom Read tritt dann ein, wenn ein Einfügungs- oder Löschvorgang in einer Zeile ausgeführt wird, die zu einem Zeilenbereich gehört, der von einer Transaktion gelesen wird. Der erste Lesevorgang der Transaktion für den Zeilenbereich zeigt eine Zeile, die im darauf folgenden Lesevorgang nicht mehr vorhanden ist, weil sie von einer anderen Transaktion gelöscht wurde. Es ist aber auch möglich, dass als Folge eines Einfügevorgangs durch eine andere Transaktion bei einem Folgelesevorgang einer Transaktion eine Zeile angezeigt wird, die im ursprünglichen Lesevorgang nicht vorhanden war.

[26] [MSOD5]

4.3.1 Isolationsstufen in SQL Server

Der ANSI SQL:1999-Standard definiert die folgenden Isolationsstufen, die alle von dem Microsoft SQL Server Datenbankmodul unterstützt werden:

- **Read Uncommitted:** Gibt an, dass Anweisungen Zeilen lesen können, die von anderen Transaktionen geändert wurden, für die jedoch noch kein COMMIT ausgeführt wurde.
- **Read Committed:** Gibt an, dass Anweisungen Zeilen nicht lesen können, die von anderen Transaktionen geändert wurden, für die jedoch noch kein COMMIT ausgeführt wurde. Dadurch werden Dirty Reads verhindert. Daten können von anderen Transaktionen zwischen einzelnen Anweisungen innerhalb der aktuellen Transaktion geändert werden, was zu Non Repeatable Reads oder Phantom Reads führt. Diese Option ist die Default-Isolationsstufe von SQL Server.
- **Repeatable Read:** Gibt an, dass Anweisungen keine Daten lesen können, die geändert wurden, für die jedoch noch kein COMMIT von anderen Transaktionen ausgeführt wurde. Darüber hinaus können von der aktuellen Transaktion gelesene Daten erst nach Abschluss der aktuellen Transaktion von anderen Transaktionen geändert werden.
- **Serializable gibt folgendes an:**
    - Anweisungen können keine Daten lesen, die geändert wurden, für die jedoch noch kein COMMIT von anderen Transaktionen ausgeführt wurde.
    - Andere Transaktionen können Daten, die von der aktuellen Transaktion gelesen werden, erst dann ändern, wenn die aktuelle Transaktion abgeschlossen ist.
    - Andere Transaktionen können erst nach Abschluss der aktuellen Transaktion neue Zeilen mit Schlüsselwerten einfügen, die in den von Anweisungen in der aktuellen Transaktion gelesenen Schlüsselbereich fallen.

SQL Server 2005 unterstützt außerdem zwei Transaktionsisolationsstufen, bei denen die Zeilenversionsverwaltung unterstützt wird. Eine davon ist eine neue Implementierung der Read Committed-Isolation, die andere – Snapshot – ist eine völlig neue Transaktionsisolationsstufe:

- Wenn die READ_COMMITTED_SNAPSHOT-Datenbankoption auf ON gesetzt ist, verwendet die Read-Committed-Isolation die Zeilenversionsverwaltung, um eine Lesekonsistenz auf der Anweisungsebene zu gewährleisten. Lesevorgänge erfordern dabei lediglich SCH-S-Sperren auf der Tabellenebene und keine Seiten- oder Zeilensperren. Wenn die READ_COMMITTED_SNAPSHOT-Datenbankoption auf OFF gesetzt ist, was der Standardeinstellung entspricht, verhält sich die Read Committed-Isolation wie in früheren Versionen von SQL Server. Beide Implementierungen entsprechen der ANSI-Definition der Read Committed-Isolation.
- **Snapshot:** Gibt an, dass von Anweisungen in einer Transaktion gelesene Daten der im Hinblick auf Transaktionen konsistenten Version der Daten entsprechen, die zu Beginn der Transaktion vorhanden waren. Die Transaktion kann nur Datenänderungen erkennen, für die vor dem Beginn der Transaktion ein COMMIT ausgeführt wurde. Datenänderungen, die nach Beginn der aktuellen Transaktion von anderen Transaktionen vorgenommen wurden, sind für in der aktuellen Transaktion ausgeführte Anweisungen nicht sichtbar. So entsteht der Eindruck, als ob die Anweisungen in einer Transaktion einen Snapshot der Daten erhalten, für die ein COMMIT ausgeführt wurde, wie sie zu Beginn der Transaktion vorhanden waren.

Die folgende Tabelle veranschaulicht, welche Parallelitätsnebeneffekte in den einzelnen Isolationsstufen zulässig sind:

Parallelitätsnebeneffekte in SQL Server

Isolationsstufe	Dirty Read	Non Repeatable Read	Phantom Read
Read Uncommitted	Ja	Ja	Ja
Read Committed	Nein	Ja	Ja
Repeatable Read	Nein	Nein	Ja
Snapshot	Nein	Nein	Nein
Serializable	Nein	Nein	Nein

Tabelle 17: Parallelitätsnebeneffekte in SQL Server

4.3.2 Isolationsstufen in Oracle

Oracle unterscheidet die zwei Isolationsstufen Read Committed und Serializable für seine Transaktionen, wobei auch hier Read Committed die Default-Isolationsstufe ist.
Die folgende Tabelle veranschaulicht, welche Parallelitätsnebeneffekte in den beiden Isolationsstufen zulässig sind:

Parallelitätsnebeneffekte in Oracle

Isolationsstufe	Dirty Read	Non Repeatable Read	Phantom Read
Read Committed	Nein	Ja	Ja
Serializable	Nein	Nein	Nein

Tabelle 18: Parallelitätsnebeneffekte in Oracle

Weiterhin ist noch über die Flashback-Funktionalität ein Pepeatable Read möglich.

5 Migration von Oracle zu SQL Server

Nachfolgend wird eine Migration von Oracle zu SQL Server anhand eines Praxisbeispiels demonstriert. Die Migration geschieht mit Hilfe des Migrationswerkzeuges Microsoft Migration Assistant (SSMA for Oracle) Version 3.0.764[27], die kostenlos von der Microsoft-Seite heruntergeladen werden kann.

In der folgenden Abbildung wird das ER-Modell einer Beispieldatenbank “DEMO“ dargestellt, dass auf eine Oracle 10g Datenbank aufgesetzt und für die Migration zu Microsoft SQL Server 2005 verwendet wurde. Alle in diesem Abschnitt abgebildeten Codezeilen und erstellten Reports basieren auf dieser Beispieldatenbank. Das ER-Modell wurde mit Hilfe des Toad Data Modelers[28] durch ein Reverse Engineering erstellt:

[27] [SSMA]
[28] [ToadData]

Toad Data Modeller:

ER-Modell Oracle-Beispieldatenbank “DEMO“

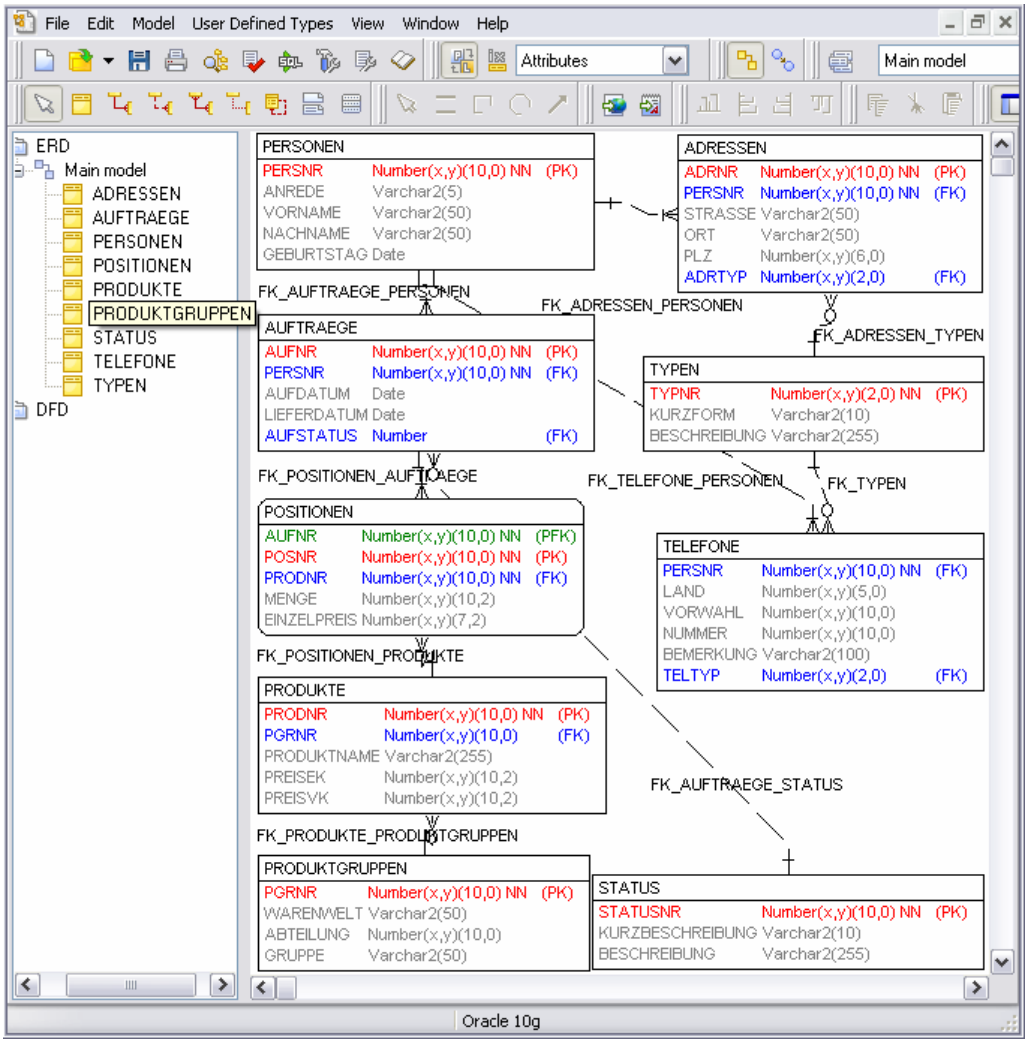


Abbildung 1: Snapshot TDM mit ER-Modell Oracle-DEMO

5.1 Migration mit dem Microsoft SQL Server Migration Assistant (SSMA for Oracle)

Der SSMA for Oracle unterstützt die automatisierte Migration von Datenbanken von Oracle zu Microsoft SQL Server. Zusammengefasst erfolgt eine Migration mit dem SSMA in 5 Schritten:

1. Verbindungen mit der Quelldatenbank (Oracle) und der Zieldatenbank (SQL Server) herstellen.
2. Die Oracle-Datenbankobjekte in SQL Server-Datenbankobjekte konvertieren (Siehe Abschnitt 5.4). Diese Datenbankobjekte werden

noch nicht in die SQL Server-Instanz sondern in das SSMA Metadata geladen.

3. Assessment Report erstellen (Optional) (Siehe Abschnitt 5.2).
4. Die konvertierten SQL Server-Datenbankobjekte in die SQL Server Datenbank laden.
5. Alle Tabellenwerte durch einen Massenimport in die SQL Server-Tabellen migrieren, wobei die Datenzeilen als Transaktionen von den Oracle-Tabellen in die SQL Server-Tabellen verschoben werden.

Die folgende Abbildung zeigt welche verschiedenen Migrationsfenster der Benutzer bei einer Migration mit dem SSMA sieht, wobei nicht konvertierbarer Code deutlich markiert wird:

Microsoft SQL Server Migration Assistant for Oracle

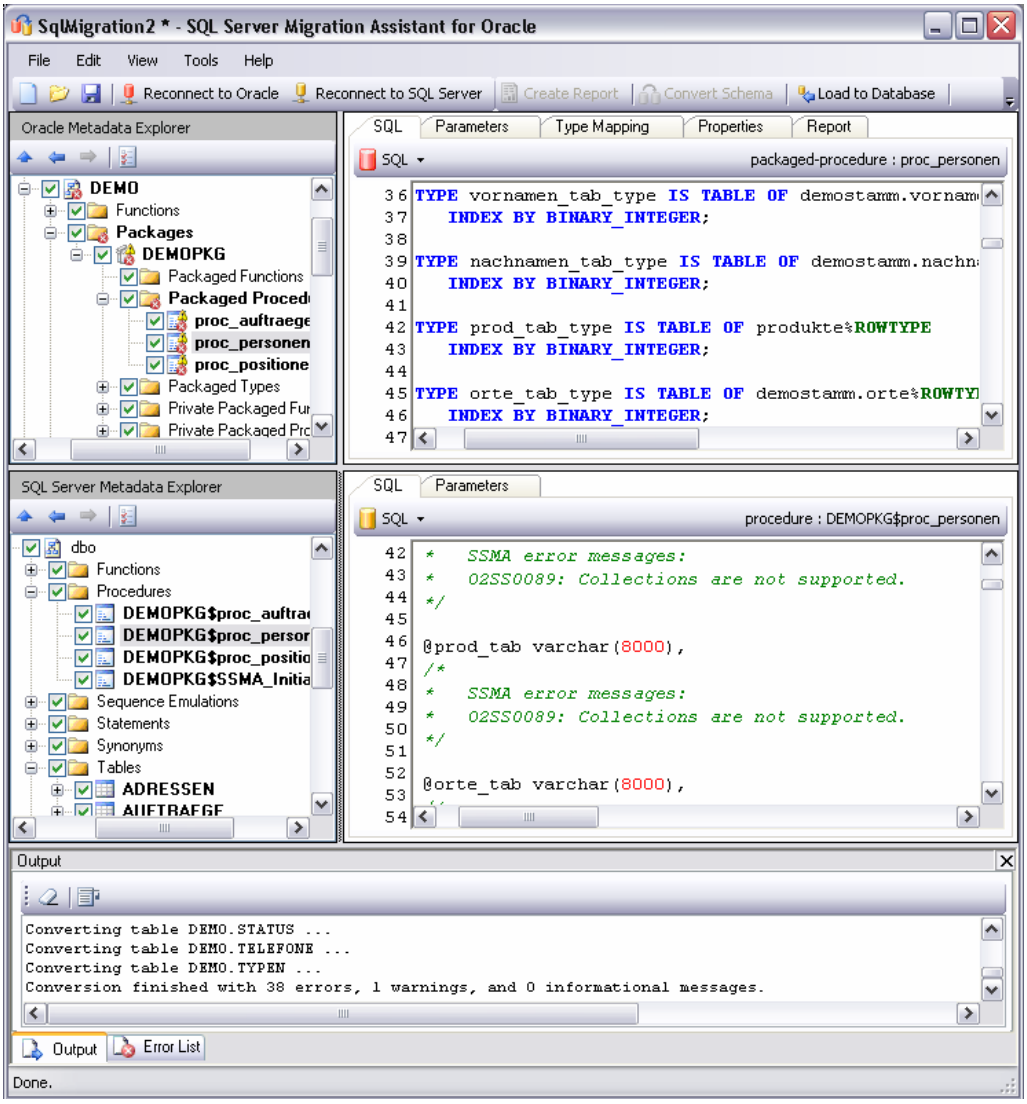


Abbildung 2: Snapshot des Microsoft Migration Assistant for Oracle

5.2 Assessment Report

Der Assessment Report enthält quantitative und qualitative Bewertungen in Form von statistischen Daten wie Gesamtzahl der Datenbankobjekte, Anzahl der Codezeilen, geschätzte Komplexität der Konvertierung sowie geschätzter Aufwand an manueller Nachbearbeitung.

Die folgende Konvertierungsstatistik wurde von SSMA für die Beispieldatenbank erstellt. Die mit einem roten X gekennzeichneten Felder geben bekannt, dass es nicht möglich ist ein PL/SQL Package für die Bewegungstabellen Fehlerfrei zu konvertieren:

Konvertierungsstatistik des SSMA

Statement type	Total	Converted
ALL	272	87.13%
argument	10	100%
assignment-statement	46	78.26%
block-statement	5	100%
column	42	100%
commit-work-statement	7	100%
create-statement	3	100%
exception-block	3	100%
exit-statement	1	100%
fetch-statement	1	100%
foreign-key	17	94.11%
for-statement	12	100%
if-statement	17	94.11%
insert-statement	5	0%
loop-statement	1	100%
open-statement	2	0%
package	1	100%
primary-key	8	100%
private-packaged-type	1	0%
private-packaged-variable	1	100%
procedure-call	12	91.66%
select-statement	15	100%
sequence	3	100%
type-declaration	8	12.5%
variable-declaration	51	86.27%

Tabelle 19: Konvertierungsstatistik des SSMA

Der Assessment Report stellt auf einer Konsolenhälfte die Datenbankobjekte der Quelldatenbank und auf der anderen Konsolenhälfte die übersetzten Datenbankobjekte, die zum Einfügen in die Zieldatenbank bereit sind, dar. Dabei wird nicht in T-SQL Code konvertierbarer PL/SQL Code als Fehler (Error) markiert:

Assessment Report

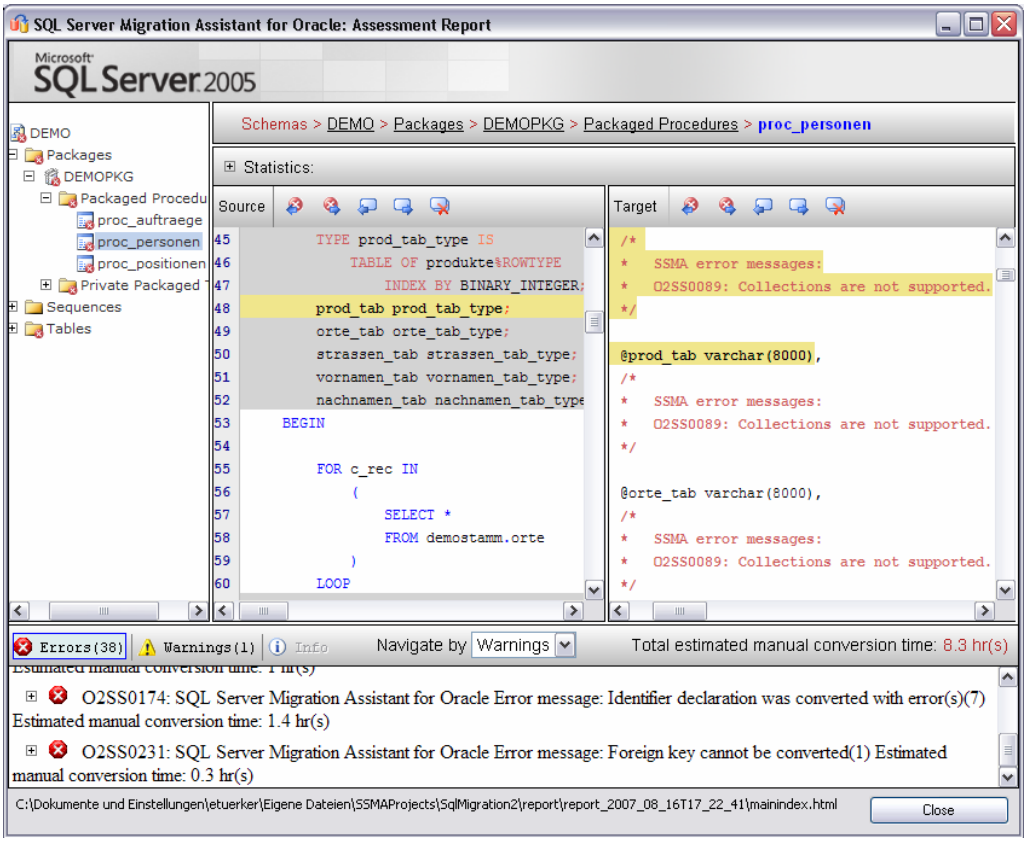


Abbildung 3: Snapshot eines Assessment Reports

5.3 Probleme

Das PL/SQL Package, zur Generierung der Bewegungstabellen für die Beispieldatenbank, erzeugte Errors und konnte von SSMA nicht in syntaktisch korrekten T-SQL Code konvertiert werden und musste manuell nachbearbeitet werden. Beispielhaft sollen hier einige Codeabschnitte der Prozedur “proc_personen“ dargestellt werden:

Collections

Da Collections in T-SQL nicht unterstützt werden, konnte diese Collection (auch indizierte Tabelle genannt), mit deren Hilfe eine geordnete Gruppe von Elementen desselben Typs erzeugt wird, nicht in T-SQL Code konvertiert werden:

PL/SQL

```
TYPE vornamen_tab_type IS TABLE OF vornamen%ROWTYPE
      INDEX BY BINARY_INTEGER;

vornamen_tab      vornamen_tab_type;
```

T-SQL

```
/*
*     SSMA error messages:
*     O2SS0089: Collections are not supported.
*/

@vornamen_tab varchar(8000)
```

Da nun die Collection nicht aufgelöst werden konnte, werden in allen Codeabschnitten Errors gemeldet, die `vornamen_tab` oder andere Collections enthalten:

PL/SQL

```
INSERT INTO personen
      (persnr, anrede, vorname, nachname, geburtstag)
VALUES
      (l persnr, vornamen tab (vornamennummer).anrede,
       vornamen_tab (vornamennummer).vorname, nachnamen_tab
      (nachnamennummer).nachname, l_geburtstag);
```

T-SQL

```
/*
*   SSMA error messages:
*   O2SS0083: Identifier vornamen_tab.anrede cannot be
*   converted because it was not resolved.
*   O2SS0083: Identifier vornamen_tab.vorname cannot be
*   converted because it was not resolved.
*   O2SS0083: Identifier nachnamen_tab.nachname cannot be
*   converted because it was not resolved.

INSERT dbo.PERSONEN
      (PERSNR, ANREDE, VORNAME, NACHNAME, GEBURTSTAG)
VALUES
      (@l_persnr, vornamen_tab.anrede, vornamen_tab.vorname,
       nachnamen_tab.nachname, @l_geburtstag)
*/
```

Korrektur

Um die Collection nachzubilden wurde eine Table-Variable `@vornamen_tab` erstellt. Die Tabellendeklaration schließt Spaltendefinitionen, Namen, Datentypen und Einschränkungen aus der Vornamen-Tabelle ein:

T-SQL

```
DECLARE @vornamen_tab TABLE (
      [ARTIFICIAL_INDEX4] [numeric](10, 0) NOT NULL,
      [VORNAMENNR] [numeric](10, 0) NOT NULL ,
      [ANREDE] [varchar](255) NULL ,
      [VORNAME] [varchar](255) NULL)
```

Die benötigten Werte können nun aus den jeweiligen Table-Typen (`@vornamen_tab`, `@nachnamen_tab`, …) ausgelesen und in die Tabelle Personen eingefügt werden:

T-SQL

```
SELECT @local_anrede = anrede FROM @vornamen_tab
WHERE artificial_index4 = @vornamennummer

SELECT @local_vorname = vorname FROM @vornamen_tab
WHERE artificial_index4 = @vornamennummer

SELECT @local_nachname = nachname FROM @nachnamen_tab
WHERE artificial_index5 = @nachnamennummer

INSERT dbo.PERSONEN
      (PERSNR, ANREDE, VORNAME, NACHNAME, GEBURTSTAG)
VALUES
      (@l_persnr, @local_anrede, @local_vorname,
       @local_nachname, @l_geburtstag)
```

Da, wie in der vorigen Situation, die Collection nicht aufgelöst werden konnte, erzeugt die Verwendung der Collection `vornamen_tab` auch in diesen Codeabschnitten einen Error. Auffallend ist dabei, dass für eine 5 Zeilen lange PL/SQL Schleife ein über 40 Zeilen langer T-SQL Code erstellt werden muss:

PL/SQL

```
FOR c_rec IN (SELECT * FROM demostamm.vornamen)

LOOP
      vornamen_tab (vornamen_tab.COUNT + 1) := c_rec;
END LOOP;
```

T-SQL

```
DECLARE @v_DB_IMPLICIT_CURSOR_FOR_c_rec_rowcount$4 int,
        @c_rec$4 xml

DECLARE DB_IMPLICIT_CURSOR_FOR_c_rec CURSOR LOCAL FOR
SELECT VORNAMEN.VORNAMENNR, VORNAMEN.ANREDE, VORNAMEN.VORNAME
FROM DEMOSTAMM.dbo.VORNAMEN

SET @v_DB_IMPLICIT_CURSOR_FOR_c_rec_rowcount$4 = 0

OPEN DB_IMPLICIT_CURSOR_FOR_c_rec

WHILE 1 = 1

   BEGIN
```

```
      DECLARE @c_rec$vornamennr float(53)       DECLARE
@c_rec$anrede varchar(8000)

       DECLARE @c_rec$vorname varchar(8000)

       FETCH DB_IMPLICIT_CURSOR_FOR_c_rec
          INTO @c_rec$vornamennr, @c_rec$anrede, @c_rec$vorname

       IF @@FETCH_STATUS = 0
          SET @v_DB_IMPLICIT_CURSOR_FOR_c_rec_rowcount$4 =
@v_DB_IMPLICIT_CURSOR_FOR_c_rec_rowcount$4 + 1

          SET @c_rec$4 =
sysdb.ssma_oracle.SetRecord_varchar(@c_rec$4, N'VORNAME',
@c_rec$vorname)

          SET @c_rec$4 =
sysdb.ssma_oracle.SetRecord_varchar(@c_rec$4, N'ANREDE',
@c_rec$anrede)

          SET @c_rec$4 =
sysdb.ssma_oracle.SetRecord_float(@c_rec$4, N'VORNAMENNR',
@c_rec$vornamennr)

          IF @@FETCH_STATUS =  -1
            BREAK

          /*
          *   SSMA error messages:
          *   O2SS0174: The declaration of the identifier
          *   '@vornamen_tab' was converted with error(s).

          SET @vornamen_tab = @c_rec$4
          */

   END

CLOSE DB_IMPLICIT_CURSOR_FOR_c_rec

DEALLOCATE DB_IMPLICIT_CURSOR_FOR_c_rec
```

Korrektur

Anstatt eine CURSOR-FOR-Schleife zu verwenden um die Vornamen-Tabelle auszulesen, werden die zuvor definierten CURSOR-Variablen verwendet um den Inhalt der Vornamen-Tabelle in die zuvor definierte (Seite 43) Table-Variable `@vornamen_tab` einzufügen:

T-SQL

```
DECLARE @temp_cnt4 numeric(10,0)
SELECT @temp_cnt4=COUNT(*) FROM @vornamen_tab
INSERT INTO @vornamen_tab VALUES
(@temp_cnt4+1,@c_rec$vornamennr, @c_rec$anrede, @c_rec$vorname)
```

Prozedur DBMS_LOCK.sleep

Da T-SQL das PL/SQL Paket DBMS_LOCK nicht besitzt kann die Prozedur `DBMS_LOCK.sleep` nicht nachgebildet werden:

PL/SQL

```
IF (i_sleep IS NOT NULL)
THEN
      DBMS_LOCK.sleep (i_sleep);
END IF;
```

T-SQL

```
IF (@i_sleep IS NOT NULL)
/*
*   SSMA error messages:
*   O2SS0083: Identifier DBMS_LOCK.sleep cannot be converted
*   because it was not resolved.

    EXECUTE DBMS_LOCK.sleep
*/

DECLARE @db_null_statement int
```

Korrektur

Mit der WAITFOR-Anweisung ist auch eine Verzögerung möglich:

T-SQL

```
IF (@i_sleep IS NOT NULL)
      WAITFOR DELAY @i_sleep
```

Datumskonvertierung

Die Spalte Geburtstag mit dem Oracle-Datentyp DATE der Tabelle Personen wurde in den SQL Server-Datentyp DATETIME konvertiert:

Tabelle Personen in Oracle

PERSNR	ANREDE	VORNAME	NACHNAME	GEBURTSTAG
1	Herr	Johann	Emmioglu	11.08.1945
2	Frau	Alice	Ahrends	10.11.1963
3	Herr	Henry	Kube	21.01.1961
4	Frau	Brigitte	Fickert	21.05.1980
5	Frau	Mareike	Netzko	04.01.1981
6	Herr	Gojko	Bernsen	09.08.1954
7	Herr	Petar	Drazen	07.03.1978
8	Frau	Lieselotte	Gerchel	19.07.1957
9	Frau	Gudrun	Baum	01.08.1965
10	Herr	Jakob	Meier	12.05.1947

Abbildung 4: Tabelle Personen in Oracle

Da der Oracle-Datentyp DATE zusätzlich noch die Uhrzeit enthält wird es auch so von dem SQL Server-Datentyp DATETIME umgesetzt. Nun wird die Spalte Geburtstag von der Tabelle Personen in SQL Server noch zusätzlich mit einer Uhrzeit angezeigt:

Tabelle Personen in SQL Server

PERSNR	ANREDE	VORNAME	NACHNAME	GEBURTSTAG
1	Herr	Pius	Meyer	26.11.1967 00:00:00
2	Herr	Horst	Seifried	26.09.1948 00:00:00
3	Herr	Henri	Ningel	15.05.1947 00:00:00
4	Herr	Enno	Emons-Impex	20.08.1957 00:00:00
5	Frau	Fritz	Behns	24.08.1983 00:00:00
6	Frau	Katrin	Schultze	02.01.1959 00:00:00
7	Herr	Karl	Otto	20.01.1948 00:00:00
8	Frau	Marlies	Mahnhardt	18.08.1944 00:00:00
9	Herr	Lothar	Körte	04.11.1952 00:00:00
10	Frau	Anneliese	Baum	19.10.1976 00:00:00

Abbildung 5: Tabelle Personen in SQL Server

5.4 Konvertierungsweise

Die folgende Tabelle zeigt wie der SSMA Oracle Datenbankobjekte in SQL Server Datenbankobjekte konvertiert[29]. Dabei verwendet der SSMA größtenteils die in Abschnitt 3.2 beschriebenen Syntax-Zuordnungen, die hier nicht noch einmal wiederholt werden sollen. Die hier genannte Konvertierungsweise des SSMA kann unter den Projekteinstellungen (Projekt Settings) geändert werden, wobei dann aber der nicht konvertierbare Code durch Fehlermeldungen (Errors) markiert wird:

Konvertierungsweise des SSMA

Oracle Datenbank-objekte	Resultierende SQL Server Datenbankobjekte
Funktionen	Wenn die Funktion direkt in T-SQL konvertiert werden kann, dann erzeugt der SSMA eine Funktion. In manchen Fällen werden Oracle-Funktionen aber als autonome Transaktionen definiert oder beinhalten Anweisungen, die in SQL Server nicht gültig sind. In diesen Fällen erzeugt der SSMA eine **Gespeicherte Prozedur** und eine **Wrapper-Funktion**, die die implementierte Gespeicherte Prozedur aufruft.
Prozeduren	Wenn die Funktion direkt in T-SQL konvertiert werden kann, dann erzeugt der SSMA eine Stored Procedure. In manchen Fällen muss eine Stored Procedure aber in einer autonomen Transaktion aufgerufen werden. In diesen Fällen erzeugt der SSMA zwei **Gespeicherte Prozeduren**, wobei eine die Prozedur implementiert und die andere zum Aufruf der implementierten Gespeicherte Prozedur dient.
Pakete	Der SSMA erzeugt ein **Set von Gespeicherte Prozeduren** und **Funktionen**, die durch ähnliche Objektnamen vereinigt werden.
Sequenzen	Der SSMA imitiert Oracle-Sequenzen.
Tabellen und Views mit abhängigen Datenbank-objekten wie z. B. Indizes und Trigger	Der SSMA erzeugt Tabellen und Views mit abhängigen Datenbankobjekten.

[29] [MA_HLP2]

Synonyme	Für die folgenden Objekttypen werden **Synonyme** erzeugt: • Tabellen und Objekttabellen • Views und Objektviews • Gespeicherte Prozeduren • Funktionen • Materialisierte Views Für die folgenden Objekttypen werden **Synonyme** aufgelöst und durch **direkte Objektreferenzen** ersetzt: • Sequenzen • Pakete • Schema-Objekte von Javaklassen • Benutzerdefinierte Objekttypen Andere Synonyme können nicht migriert werden. Der SSMA markiert diese Synonyme und alle Referenzen, die diese Synonyme verwenden als Errors. Die Projekt Settings können so eingestellt werden, dass alle hier genannten Synonyme durch direkte Objektreferenzen ersetzt werden.
COUNT	Der SSMA konvertiert sicherheitshalber alle **COUNT**-Funktionen in **COUNT_BIG**, damit auch Werte größer als 2147483647 (2^{31}-1) zurückgegeben werden können.
SUBSTR	Der SSMA konvertiert Oracle **SUBSTR** Funktionsaufrufe in SQL Server **SUBSTRING** Funktionsaufrufe, in Abhängigkeit von der Anzahl der Parameter. Falls ein SUBSTR Funktionsaufruf nicht konvertiert oder die Anzahl der Parameter nicht ermittelt werden kann, wird dieser in einen spezifischen SSMA Funktionsaufruf konvertiert.
Transaction Processing	Oracle Transaction Processing-Anweisungen werden in SQL Server-Anweisungen konvertiert. Oracle öffnet Transaktionen implizit. Um dieses Verhalten auf dem SQL Server zu imitieren muss eine **BEGIN TRANSACTION-Anweisung** an den Stellen manuell eingefügt werden, wo die Transaktion starten soll. Alternativ kann noch die **SET IMPLICIT_TRANSACTIONS ON-Anweisung** am Session-Anfang ausgeführt werden. Der SSMA führt **SET IMPLICIT_TRANSACTIONS ON-Anweisungen** automatisch bei der Konvertierung von Subroutinen mit autonomen Transaktionen aus.

Tabelle 20: Konvertierungsweise des SSMA

ROWNUM

Der SSMA konvertiert ROWNUM-Spalten, die in Oracle zur Einschränkung der Ergebnismenge dienen, in eine TOP-Klausel auf die der Ausdruck folgt. Das folgende Beispiel zeigt ROWNUM in einer SELECT-Anweisung:

```
SELECT FROM Table1
WHERE ROWNUM < expression and Field1 >= 2
```

Das folgende Beispiel zeigt den daraus resultierenden T-SQL Code:

```
SELECT TOP (expression-1)
FROM Table1
WHERE Field1>=2
```

Mit Hilfe der TOP-Klausel kann die Anzahl der Zeilen, die in einer SELECT-Anweisung ausgelesen werden sollen, beschränkt werden. Der TOP-Klausel folgende Ausdruck wird zu einem Integerwert evaluiert. Falls der Integerwert negativ ist, erzeugt die Anweisung einen Error.

ROWID

Wenn von SSMA Tabellen in SQL Server erzeugt werden, können auch ROWID-Spalten erzeugt werden. Wenn die Daten eingefügt werden, erhält jede Zeile einen neuen **UNIQUEIDENTIFIER-Wert**, der von einer newid()-Funktion erzeugt wurde. Der UNIQUEIDENTIFIER -Datentyp speichert aus 16 Bytes bestehende Binärwerte, die als **GUIDs** (Globally Unique Identifier) fungieren. Ein GUID ist eine eindeutige Binärzahl.

Die Projekt Settings können so eingestellt werden, dass für alle Tabellen ROWID-Spalten erzeugt werden und der SQL Server GUIDs erzeugt, wenn Werte eingefügt werden oder ROWID-Spalten nicht in Tabellen eingefügt werden.

Sequenz zu Identity-Konvertierung

In Oracle können Sequenzen benutzt werden um UNIQUE IDENTIFIER zu generieren. In SQL Server werden UNIQUE IDENTIFIER für Tabellen definiert, indem für die Tabelle eine Identitätsspalte erstellt wird. Eine Identitätsspalte wird meist zusammen mit der PRIMARY KEY-Einschränkung verwendet, damit sie als Zeilen-ID für die Tabelle fungiert. Der SSMA konvertiert Sequenzen, die einer Spalte zugewiesen werden, in SQL Server Identitätswerte.

Die Projekt Settings können so eingestellt werden, dass entweder eine Sequenz zu einer Identitätsspalte auf dem SQL Server table tab zugewiesen wird oder nur die SQL Server Identität und nicht die Zuweisung einer Sequenz unterstützt werden.

CURRVAL konvertieren

Da Oracle-Sequenzen von Tabellen getrennte Datenbankobjekte sind, benutzen viele Tabellen, die Sequenzen verwenden, Trigger um neue Sequenzwerte zu generieren und einzufügen. Der SSMA markiert diese Anweisungen als Errors.

5.5 Ergebnis

Nach einer manuellen Nachbearbeitung in ein syntaktisch korrektes T-SQL wurde auch die semantische (sinngemäße) Korrektheit des Codes überprüft. Dabei hat sich herausgestellt, dass der SSMA durchgehend einen semantisch korrekten Code erzeugt hat. Außer den nicht konvertierbaren Collections und der nicht konvertierbaren Prozedur `DBMS_LOCK.sleep` hat der SSMA den gesamten PL/SQL Code in T-SQL Code konvertiert.
Leider weist aber der erzeugte T-SQL Code eine sehr schlechte Performanz auf. Ein Grund dafür sind die von dem SSMA für die Konvertierung der PL/SQL Prozeduren erzeugten T-SQL Prozeduren und Wrapper-Funktion, die die implementierten T-SQL Prozeduren aufrufen. Bei der Konvertierung wird beispielsweise aus einer 234 Zeilen langen PL/SQL Prozedur nach der Konvertierung eine 622 Zeilen lange T-SQL Prozedur. Im Allgemeinen hat sich also bei der Konvertierung der ursprüngliche PL/SQL Code verdreifacht, wobei auch oft kein Native T-SQL erzeugt wurde. In dem folgenden Beispiel kann das PL/SQL Schlüsselwort `TO_CHAR` nur mittels eines “ssma_oracle-Schemas“ aus der sysdb-Datenbank konvertiert werden:

PL/SQL

```
/*--Sonntags gibt es keine Auftraege-*/

IF (TO_CHAR (l_aufdatum, 'D') > 6)
THEN
      l_aufdatum := l_aufdatum + 1;
END IF;
```

T-SQL

```
/*--Sonntags gibt es keine Auftraege-*/

IF (sysdb.ssma_oracle.to_char_date(@l_aufdatum, 'D') > 6)
      SET @l_aufdatum = @l_aufdatum + 1
```

Die sysdb-Datenbank wurde von dem SSMA vor der Konvertierung, mittels eines Extension Packs, unter SQL Server als eigene Datenbank angelegt. Es gibt noch viele weitere Beispiele in denen auf die sysdb-Datenbank zugegriffen werden muss und somit kein Native T-SQL erzeugt wird.

In der folgenden Abbildung wird das resultierende ER-Modell der zu SQL Server 2005 migrierten Beispieldatenbank dargestellt. Dabei sind, außer den Datentypen, keine weiteren Veränderungen erkennbar:

Toad Data Modeller:

ER-Modell SQL Server-Beispieldatenbank “DEMO“

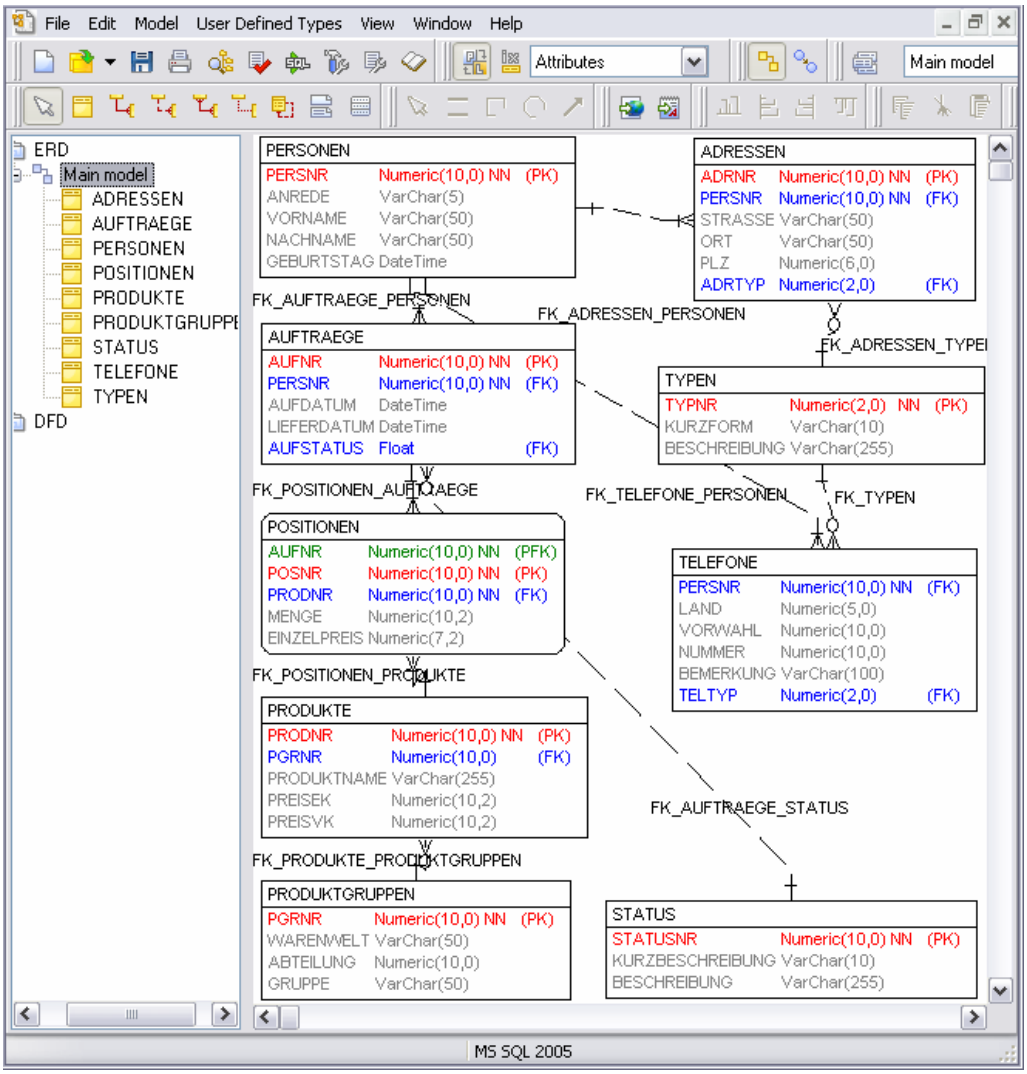


Abbildung 6: Snapshot TDM mit ER-Modell SQL Server-DEMO

6 Migration von SQL Server zu Oracle

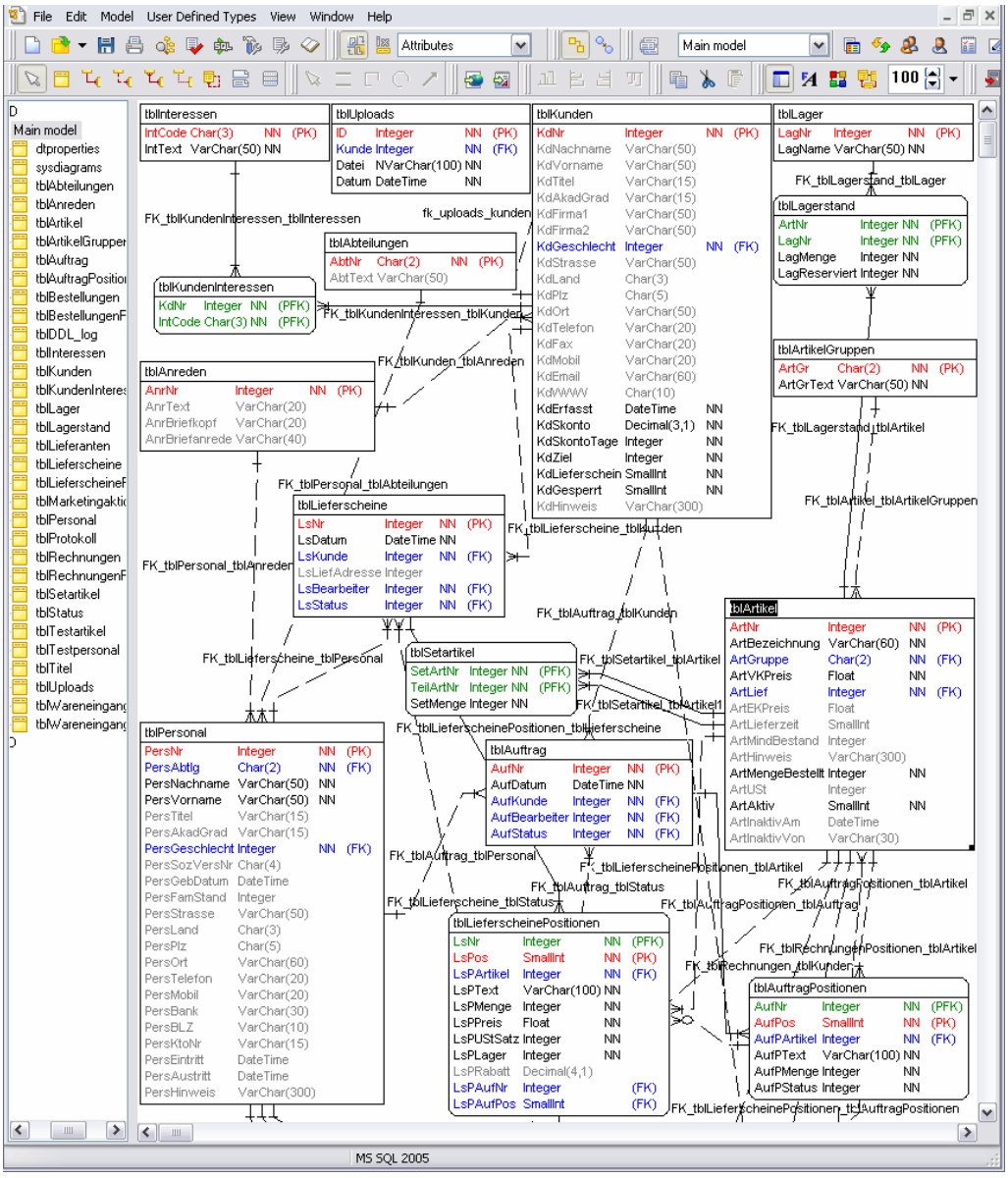
Nachfolgend wird eine Migration von SQL Server zu Oracle anhand eines Praxisbeispiels demonstriert. Die Migration geschieht mit Hilfe des Oracle SQL Developer Version 1.2.0[30], der kostenlos von der Oracle-Seite heruntergeladen werden kann.

In der folgenden Abbildung wird das ER-Modell einer Beispieldatenbank Warenwirtschaft "WAWI"[31] dargestellt, dass auf eine Microsoft SQL Server 2005 Datenbank aufgesetzt und für die Migration zu einer Oracle 10g Datenbank verwendet wurde. Alle in diesem Abschnitt abgebildeten Codezeilen basieren auf dieser Beispieldatenbank, die Tabellen, Beziehungen, Sichten, Funktionen, Trigger, Prozeduren und CLR-Code enthält:

[30] [OraDEV]
[31] [WAWI]

Toad Data Modeller:

ER-Modell SQL Server-Beispieldatenbank “WAWI“



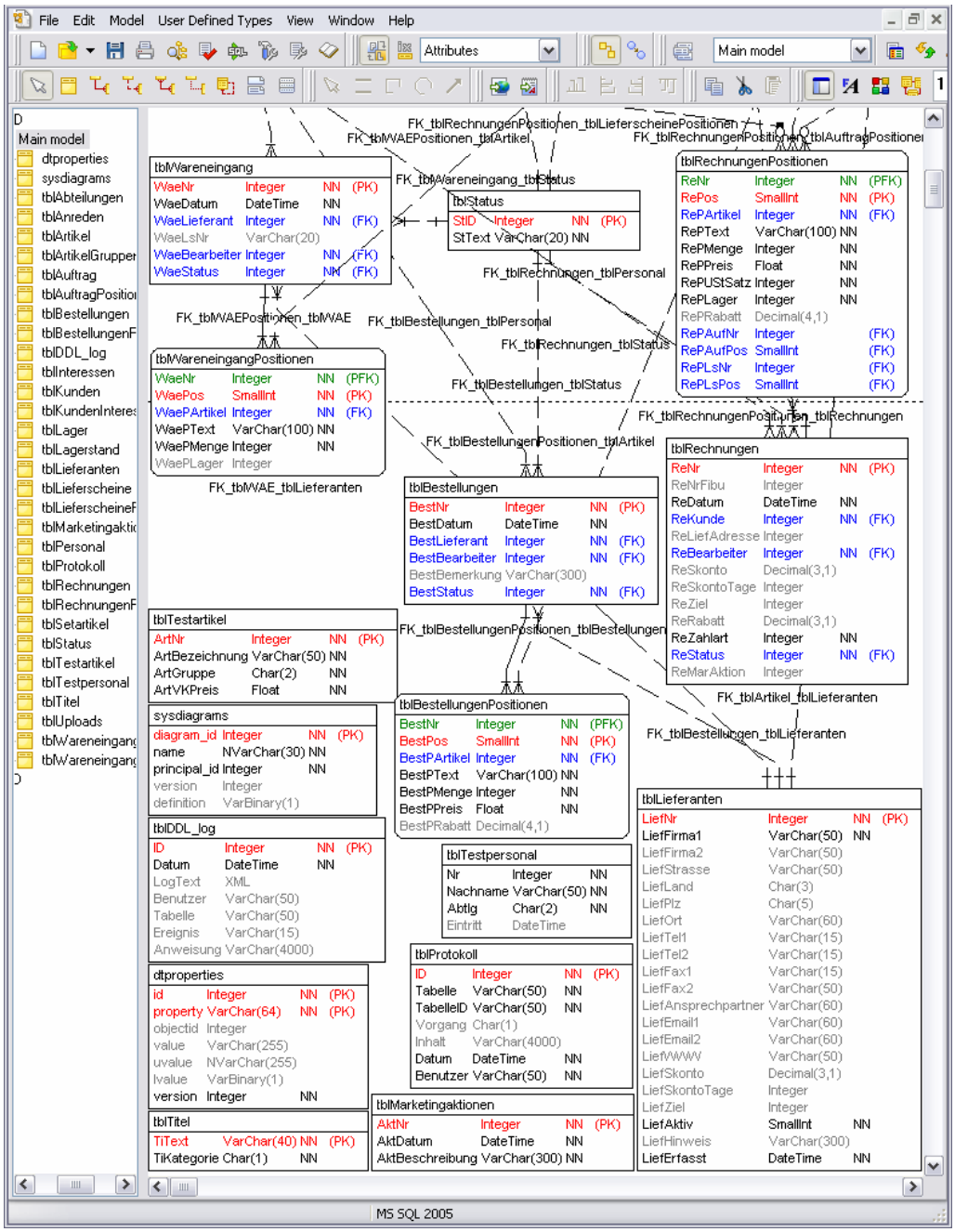


Abbildung 7: Snapshot TDM mit ER-Modell SQL Server-WAWI

6.1 Migration mit dem Oracle SQL Developer

Ursprünglich wurde die Oracle Migration Workbench von Oracle für Migrationen zur Verfügung gestellt. Für die Migration von Datenbanken von Microsoft SQL Server zu Oracle muss aber der SQL Developer verwendet werden, wobei die Migration Workbench in den SQL Developer integriert wurde. Zusammengefasst erfolgt eine Migration mit dem SQL Developer in den folgenden Schritten:

1. Verbindung mit einer zuvor in der Zieldatenbank (Oracle) erstellten Repository-Datenbank herstellen.
2. Ein Workbench Repository in der Repository-Datenbank erstellen, um die gesammelten und konvertierten Metadaten, die für den Migrationsprozess benötigt werden, zu speichern.
3. Verbindung mit der Quelldatenbank (SQL Server) herstellen.
4. Die SQL Server-Datenbank in ein Quellmodell konvertieren, dass in dem Workbench Repository gespeichert wird:
    - Dabei wird ein Snapshot der SQL Server-Datenbank erstellt, dessen gesamte Struktur erfasst wird. Die Migration Workbench arbeitet ab jetzt nur noch mit den in dem Repository gespeicherten Metadaten anstatt Abfragen gegen die laufende SQL Server-Datenbank zu erzeugen.
5. Das Quellmodell in ein Oraclespezifisches Oraclemodell konvertieren:
    - Etwa wie der SSMA übernimmt die Migration Workbench dabei SQL Server-Objektdefinitionen und konvertiert diese in die entsprechenden Oracle-Datenbankobjekte. Diese Daten werden noch nicht in die Oracle-Datenbank geladen sondern bleiben in dem Workbench Repository.
6. Script generieren für die Erzeugung der Oracle-Datenbank.
7. Verbindung mit der Oracle-Datenbank herstellen und das generierte Script ausführen, wobei alle zuvor konvertierten Oracle-Datenbankobjekte in der Oracle-Datenbank erstellt werden.
8. Alle Tabellenwerte in die Oracle-Tabellen migrieren, indem einige parallele Verbindungen aufgebaut werden um die Daten zeitgerecht abzuarbeiten.

Die folgende Abbildung zeigt, welche verschiedenen Migrationsfenster der Benutzer bei einer Migration mit dem Oracle SQL Developer sieht:

Oracle SQL Developer

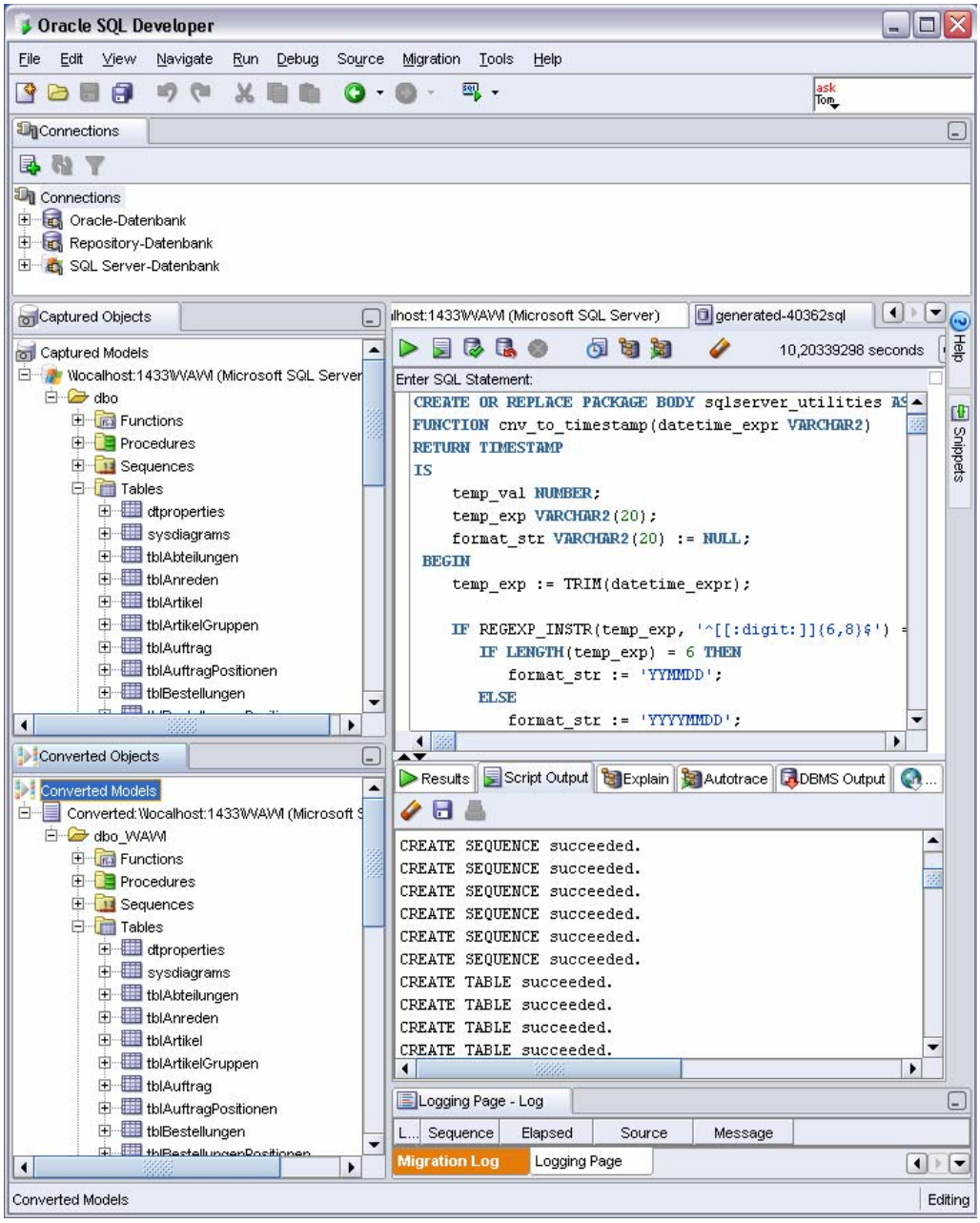


Abbildung 8: Snapshot des Oracle SQL Developers

Die folgende Abbildung verdeutlicht noch einmal wie die Migration Workbench und die Plug-Ins (Zusatzprogramme) die Informationen aus dem Source Model (Quellmodell) und dem Oracle Model lesen und damit die Oracle-Datenbank erzeugen:

Oracle Migration Workbench Architektur

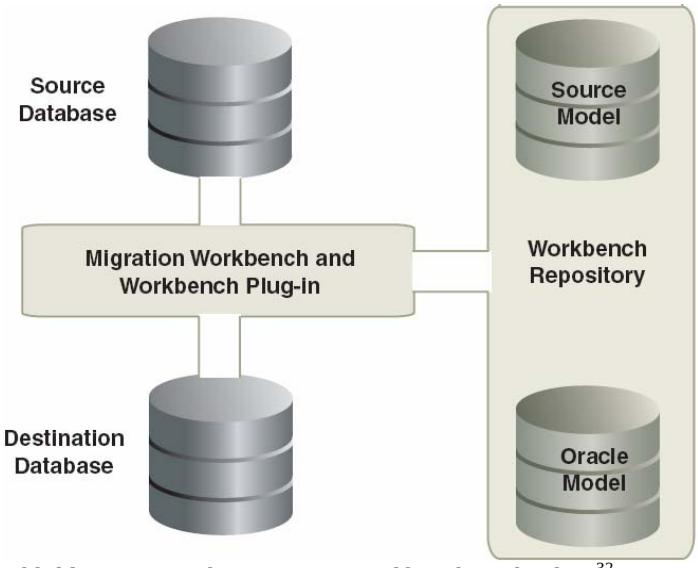


Abbildung 9: Oracle Migration Workbench Architektur[32]

6.2 Probleme

Warnungen

Bei der Ausführung des von der Migration Workbench generierten Scripts um die zuvor konvertierten Oracle-Datenbankobjekte in der Oracle-Datenbank zu erstellen werden in einem Script Output-Fenster Warnungen bei einigen Prozeduren und Funktionen ausgegeben:

SQL Developer: Script Output mit Warnungen

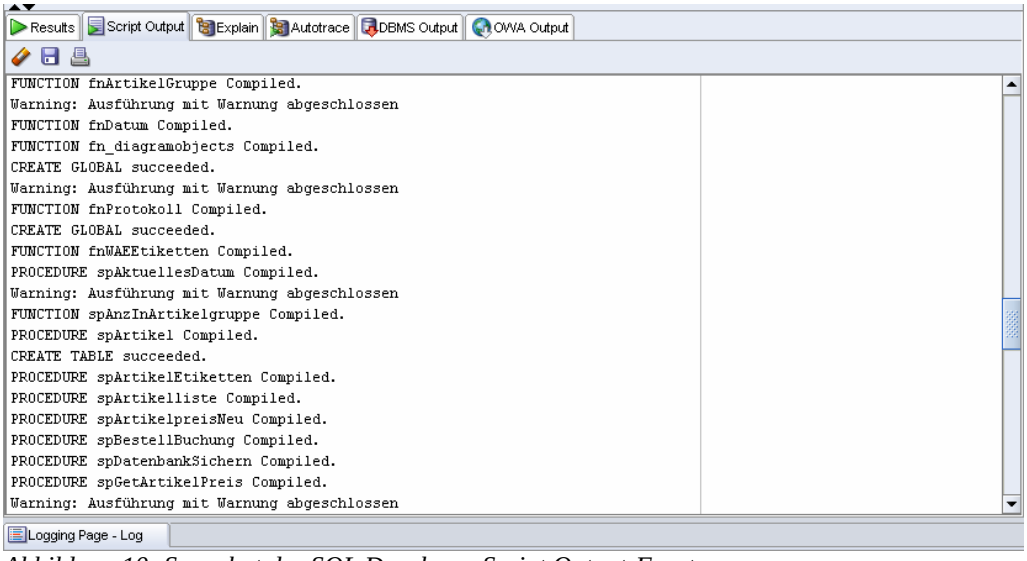


Abbildung 10: Snapshot des SQL Developer Script Output-Fensters

[32] [OraMWB]

Im Gegensatz zu SSMA (Vergleiche Abschnitt 5.1) erstellt der Oracle SQL Developer in diesen Fällen weder Markierungen in dem Quellcode noch gibt es nähere Informationen über die Ursache der Warnungen.

Errors

In den nachfolgenden konvertierten Funktionen und Triggern wurden mit Hilfe von Toad for Oracle[33] Fehler entdeckt, die aber bei der Konvertierung keinerlei Warnungen in SQL Developer erzeugten:

Toad for Oracle: Error in der Funktion fnDatum

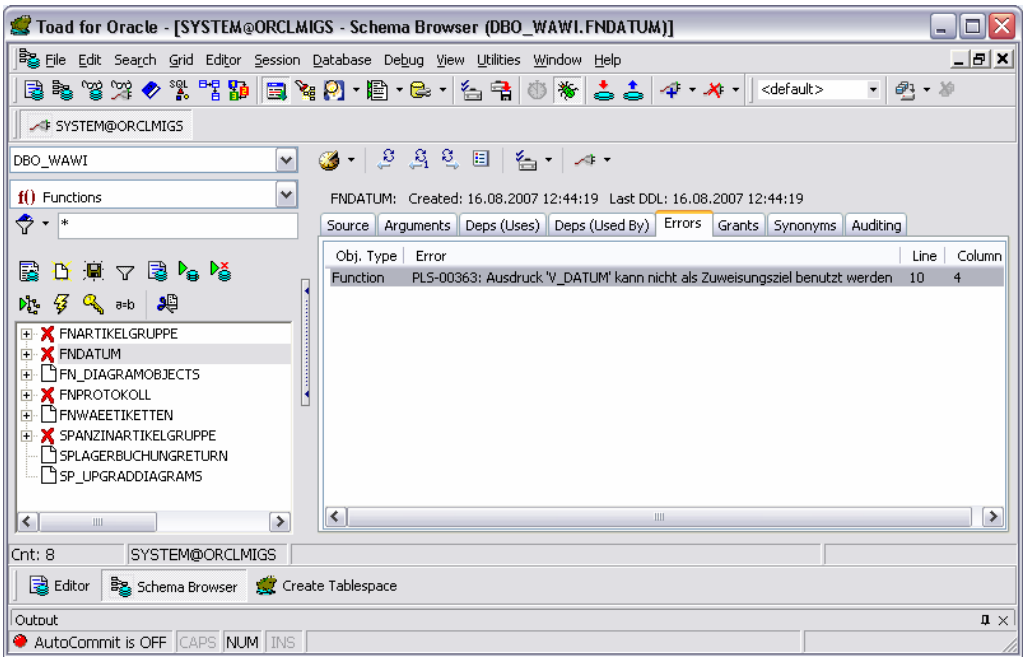


Abbildung 11: Snapshot Toad for Oracle

Funktion fnDatum

Error: `v_datum := sqlserver_utilities.convert('TIMESTAMP(6)', v_char_datum, 104);` (Ausdruck “V_DATUM” kann nicht als Zuweisungsziel benutzt werden):

[33] [ToadOra]

T-SQL

```
CREATE FUNCTION dbo.fnDatum (@datum datetime)
RETURNS datetime
AS
BEGIN

    DECLARE @char_datum varchar(10)

    SET @char_datum = CONVERT(varchar, @datum, 104)
    SET @datum = CONVERT(datetime, @char_datum, 104)

RETURN @datum
END
```

PL/SQL

```
CREATE OR REPLACE FUNCTION DBO_WAWI.fnDatum
(
  v_datum IN TIMESTAMP
)
RETURN TIMESTAMP
AS
   v char_datum VARCHAR2(10);
BEGIN
   v char datum := sqlserver_utilities.convert('VARCHAR2',
v datum, 104);
   v datum := sqlserver_utilities.convert('TIMESTAMP(6)',
v char datum, 104);
   RETURN v_datum;
END;
```

Funktion spAnzInArtikelgruppe

Fehler: `v_ergebnis :=spAnzInArtikelgruppe('GE')` (Falsche Anzahl oder Typen von Argumenten in Aufruf von SPANZINARTIKELGRUPPE):

T-SQL

```
CREATE PROCEDURE spAnzInArtikelgruppe
    @gruppe char(2)
AS
DECLARE @anz int
    SET NOCOUNT ON
    SET @anz = (SELECT COUNT(*)
                FROM tblArtikel
                WHERE ArtGruppe = @gruppe)
RETURN @anz

DECLARE @ergebnis int
EXEC @ergebnis = spAnzInArtikelgruppe 'GE'
SELECT @ergebnis As Anzahl
```

PL/SQL

```
CREATE OR REPLACE FUNCTION DBO_WAWI.spAnzInArtikelgruppe
(
  v gruppe IN CHAR DEFAULT NULL ,
  cv_1 IN OUT SYS_REFCURSOR
)
RETURN NUMBER
AS
   v anz NUMBER(10,0);
```

```
   v_ergebnis NUMBER(10,0);
BEGIN

   NULL/*TODO:SET NOCOUNT ON*/;
   SELECT COUNT(*)
     INTO v anz
     FROM tblArtikel
     WHERE ArtGruppe = v_gruppe;
   RETURN v anz;
   v ergebnis :=spAnzInArtikelgruppe('GE');
   OPEN cv 1 FOR
      SELECT v ergebnis Anzahl
        FROM DUAL ;
END;
```

Weiterhin kann, wie in dem vorigen Beispiel zu sehen ist, die SET NOCOUNT ON-Anweisung, die bewirkt, dass die Meldung bezüglich der Anzahl der von einer T-SQL Anweisung oder gespeicherten Prozedur betroffenen Zeilen nicht mehr als Teil des Resultsets zurückgegeben wird, nicht übersetzt werden. Im Gegensatz zum SSMA (Vergleiche Abschnitt 5.3) werden diese Fehler aber sehr unauffällig kommentiert und immer mit einem NULL ersetzt.

Funktion fnWAEEtiketten

Die folgende T-SQL Funktion soll ein Ergebnis in Tabellenform zurückgeben. Es werden Preisetiketten benötigt. Dabei sollen die Daten, die für den Etikettendruck benötigt werden, für jeden Artikel so oft im Ergebnis vorkommen, wie Stück gedruckte Etiketten benötigt werden.

Die nachfolgende Funktion wurde von SQL Developer falsch übersetzt und enthält eine ungewollte Endlosschleife:

T-SQL

```
CREATE FUNCTION fnWAEEtiketten (@wae int)
RETURNS @etik TABLE
    (Artikel int,
     Bezeichnung varchar(100),
     Preis smallmoney,
     Gruppe char(2))
AS
BEGIN
    DECLARE @artikel int
    DECLARE @name varchar(100), @gruppe char(2)
    DECLARE @stk int, @preis smallmoney
    DECLARE @i int

    DECLARE wae_cursor CURSOR LOCAL STATIC
    FOR
        SELECT w.WaePArtikel, w.WaePMenge, a.ArtGruppe,
a.ArtVKpreis, a.ArtBezeichnung
        FROM tblWareneingangPositionen w
        INNER JOIN tblArtikel a ON w.WaePArtikel = a.ArtNR
```

```
        WHERE w.WaeNr = @wae
    OPEN wae_cursor

    FETCH NEXT FROM wae_cursor INTO @artikel, @stk, @gruppe,
@preis, @name

    WHILE @@fetch_status = 0
    BEGIN
        SET @i = 1
        WHILE @i <= @stk
        BEGIN
            INSERT INTO @etik
            VALUES (@artikel, @name, @preis, @gruppe)

            SET @i = @i + 1
        END

        FETCH NEXT FROM wae_cursor INTO @artikel, @stk,
@gruppe, @preis, @name
    END

    CLOSE wae_cursor
    DEALLOCATE wae_cursor

RETURN
END
```

Um den CURSOR zu durchlaufen wird in der WHILE-Schleife der Zähler zu Beginn für jeden Artikel einmal auf 1 gesetzt. Dieser Zähler wird benötigt, um so lange Zeilen in die interne Tabelle einzufügen, wie Etiketten für diesen Artikel benötigt werden. Dafür wird die innere WHILE-Schleife verwendet. Bei jedem Schleifendurchlauf wird der Zähler `@i` um eins erhöht und eine Zeile in die interne Tabelle `@ethik` eingefügt. In der nachfolgenden, konvertierten PL/SQL Funktion bleibt `WHILE v_fetch_status = 0` immer 0 und wird nie verändert. Die Schleife läuft endlos weiter:

PL/SQL

```
CREATE GLOBAL TEMPORARY TABLE TT__ETIK
(
  ARTIKEL      NUMBER(10),
  BEZEICHNUNG  VARCHAR2(100 BYTE),
  PREIS        NUMBER(10,4),
  GRUPPE       CHAR(2 BYTE) );

CREATE OR REPLACE FUNCTION DBO_WAWI.fnWAEEtiketten
(
  v wae IN NUMBER,
  cv_1 IN OUT SYS_REFCURSOR
)
RETURN NUMBER
AS
   v fetch status NUMBER := 0;
   CURSOR wae cursor
     IS SELECT w.WaePArtikel,
   w.WaePMenge,
   a.ArtGruppe,
```

```
   a.ArtVKpreis,
   a.ArtBezeichnung
     FROM tblWareneingangPositionen w
     JOIN tblArtikel a
      ON w.WaePArtikel = a.ArtNR
     WHERE w.WaeNr = v wae;
   v artikel NUMBER(10,0);
   v name VARCHAR2(100);
   v gruppe CHAR(2);
   v stk NUMBER(10,0);
   v preis NUMBER(10,4);
   v i NUMBER(10,0);
BEGIN
   OPEN wae cursor;
   FETCH wae cursor INTO
v artikel,v_stk,v_gruppe,v_preis,v_name;
   CASE
      WHEN wae cursor%FOUND THEN
         v_fetch_status := 0;
      ELSE
         v fetch_status := -1;
   END CASE;

   WHILE v_fetch_status = 0
   LOOP
      BEGIN
         v i := 1;
         WHILE v_i <= v_stk
         LOOP
            BEGIN
               INSERT INTO tt  etik
                 VALUES ( v_artikel, v_name, v_preis, v_gruppe
);
               v_i := v_i + 1;
            END;
         END LOOP;
         FETCH wae cursor INTO
v_artikel,v_stk,v_gruppe,v_preis,v_name;
      END;
   END LOOP;
   CLOSE wae_cursor;
   OPEN cv 1 FOR
      SELECT *
        FROM tt__etik;

   RETURN 0;
END;
```

Prozedur spZeitbuchungSelect

Bis auf die SET NOCOUNT ON-Anweisung konnte die gesamte nachfolgende Prozedur übersetzt werden. Dabei fällt auf, dass der SQL Developer im Vergleich zum SSMA nicht so viele Codezeilen bei der Konvertierung von T-SQL in PL/SQL erzeugt:

T-SQL

```
CREATE PROCEDURE spZeitbuchungSelect

      @projekt int,
      @persnr int,
      @stunden decimal(3,1)

AS

DECLARE @stdsatz smallmoney
DECLARE @ergebnis varchar(100)

      set nocount on

      SET @stdsatz = (SELECT PersKalkStdLohn
                      FROM tblPersonal
                      WHERE PersNr = @persnr)

      IF @stdsatz Is Null
            SET @ergebnis = 'Die Personalnummer ' + @persnr + '
ist ungültig.'

      ELSE
      BEGIN

            UPDATE tblProjekte
            SET ProjStundenIst = ProjStundenIst + @stunden,
                  ProjKalkKosten = ProjKalkKosten + @stdsatz *
@stunden
            WHERE ProjNr = @projekt

            IF @@rowcount      = 1
                  SET @ergebnis = 'Zeit und Kosten erfolgreich
verbucht.'
            ELSE
                  SET @ergebnis = 'Projektnummer ungültig oder
Verbuchung nicht erfolgreich.'

      END

      SELECT @ergebnis As Ergebnis

RETURN
```

PL/SQL

```
CREATE OR REPLACE PROCEDURE spZeitbuchungSelect
(
  v projekt IN NUMBER DEFAULT NULL ,
  v persnr IN NUMBER DEFAULT NULL ,
  v stunden IN NUMBER DEFAULT NULL ,
  cv_1 IN OUT SYS_REFCURSOR
)
AS
   v stdsatz NUMBER(10,2);
   v ergebnis VARCHAR2(100);
BEGIN
   NULL/*TODO:set nocount on*/;
   SELECT PersKalkStdLohn
     INTO v_stdsatz
     FROM tblPersonal
     WHERE PersNr = v persnr;
   IF v_stdsatz IS NULL THEN
```

```
      v_ergebnis := 'Die Personalnummer ' || v_persnr || ' ist
ungültig.';
   ELSE
   BEGIN
      UPDATE tblProjekte
         SET ProjStundenIst = ProjStundenIst + v stunden,
             ProjKalkKosten = ProjKalkKosten + v_stdsatz *
v stunden
         WHERE ProjNr = v projekt;
      IF SQL%ROWCOUNT = 1 THEN
         v_ergebnis := 'Zeit und Kosten erfolgreich verbucht.';
      ELSE
         v ergebnis := 'Projektnummer ungültig oder Verbuchung
nicht erfolgreich.';
      END IF;
   END;
   END IF;
   OPEN cv 1 FOR
      SELECT v ergebnis Ergebnis
        FROM DUAL ;
   RETURN;
END;
```

6.3 Ergebnis

Obwohl der SQL Developer anfangs durch eine sehr schnelle und anscheinend fehlerfreie Konvertierung überraschte hat sich herausgestellt, dass kein durchgehend semantisch korrekter PL/SQL Code erzeugt wurde, weil beispielsweise eine Schleife in einer Funktion falsch übersetzt wurde (Siehe Abschnitt 6.2). Weiterhin wurden, im Gegensatz zum SSMA, Fehler sehr unauffällig markiert und im SQL Developer viele Warnungen ausgegeben, wobei aber die Ursache der Warnungen nicht ermittelt werden konnte. Im Allgemeinen hat sich die Codemenge nicht in dem Maße vergrößert wie bei der Konvertierung von PL/SQL in T-SQL. Im Vergleich zu dem SSMA konnte der SQL Developer auch kein durchgehendes Native PL/SQL erzeugen. In dem folgenden Beispiel kann die T-SQL Funktion `CONVERT()` nur mittels eines Packages "sqlserver_utilities“ konvertiert werden:

T-SQL

```
BEGIN

    DECLARE @char_datum varchar(10)

    SET @char_datum = CONVERT(varchar, @datum, 104)
    SET @datum = CONVERT(datetime, @char_datum, 104)

RETURN @datum
END
```

PL/SQL

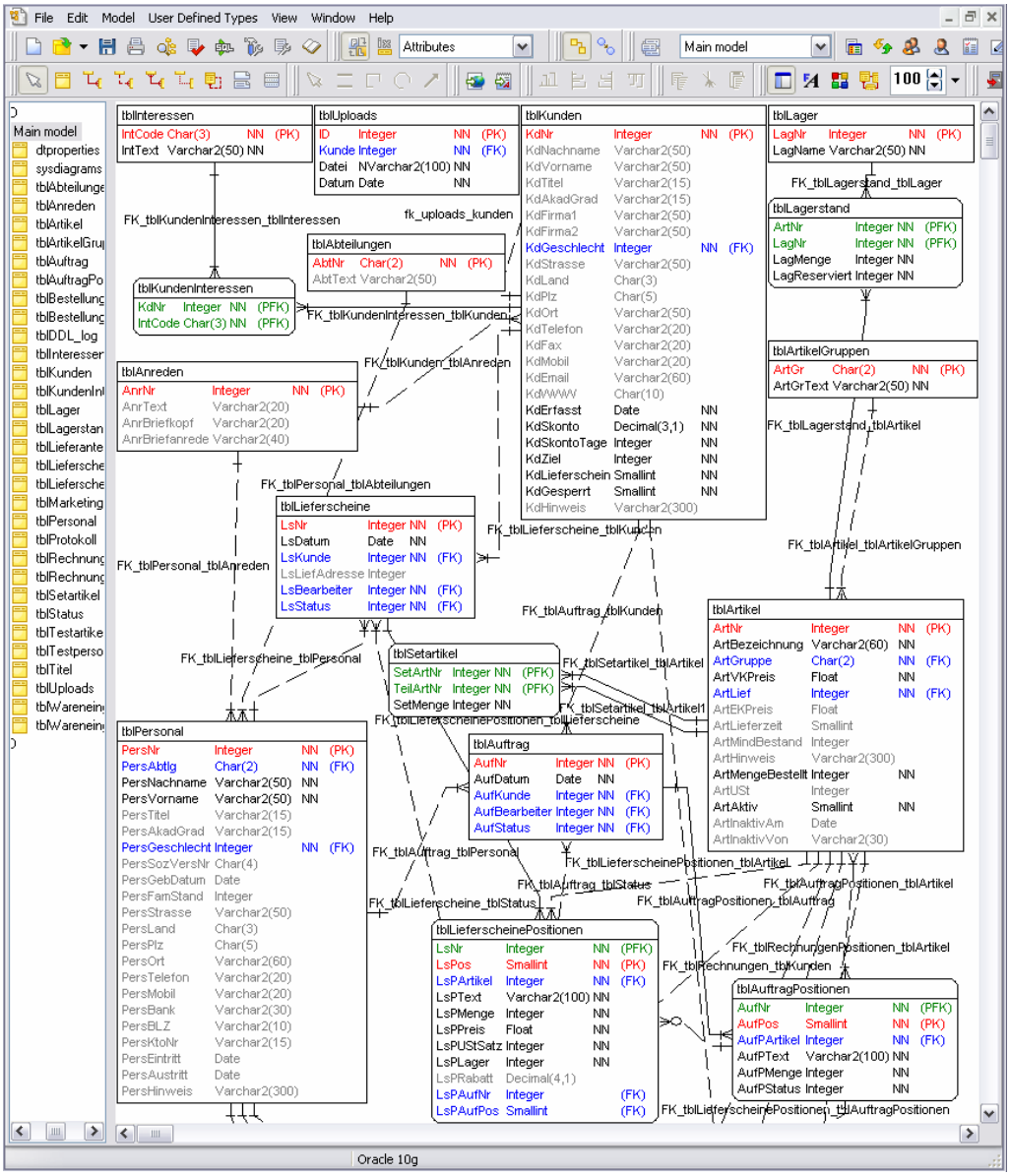
```
BEGIN
   v char datum := sqlserver_utilities.convert('VARCHAR2',
v_datum, 104);
   v datum := sqlserver_utilities.convert('TIMESTAMP(6)',
v char datum, 104);
   RETURN v_datum;
END;
```

Das Package “sqlserver_utilities“ wurde von dem SQL Developer dem User der Beispieldatenbank zugewiesen.

In der folgenden Abbildung wird das resultierende ER-Modell der zu Oracle 10g migrierten Beispieldatenbank dargestellt. Dabei sind außer den Datentypen, keine weiteren Veränderungen erkennbar. Leider konnte der Oracle SQL Developer 1.2, Beziehungen und Schlüssel aus der Quelldatenbank nicht migrieren, weil diese Version wahrscheinlich einen Bug enthält[34]. Doch mit Hilfe des Toad Data Modelers[35] konnte durch ein Reverse Engineering der ursprünglich in SQL Server erstellten Beispieldatenbank “WAWI“ und einer darauf folgenden Modell-Konvertierung in ein Oracle 10g Datenbank-Modell das folgende ER-Modell erstellt werden:

[34] [OraFRM]
[35] [ToadData]

Toad Data Modeller: ER-Modell Oracle-Beispieldatenbank “WAWI“



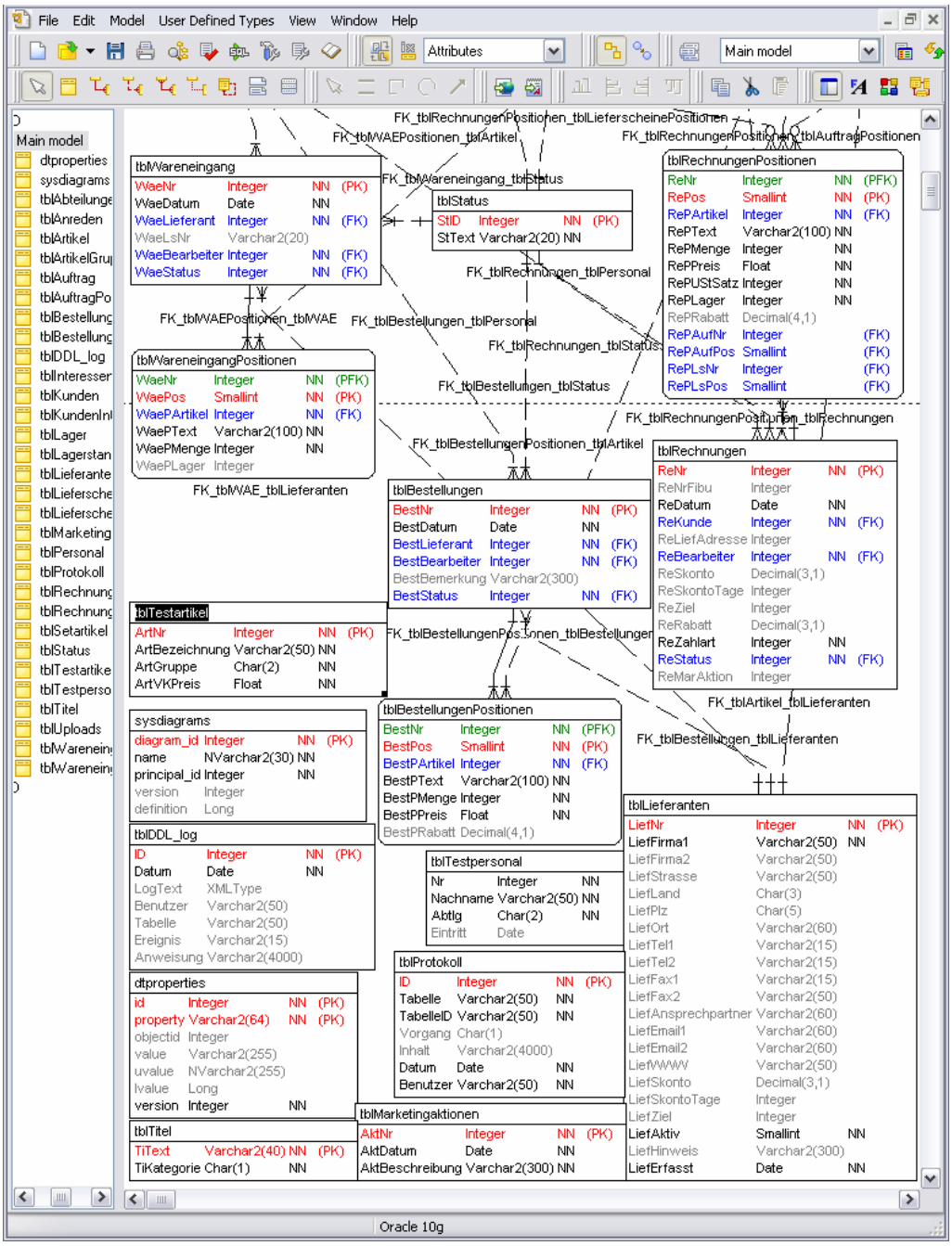


Abbildung 12: Snapshot TDM mit ER-Modell Oracle-WAWI

7 Schlussbetrachtung

Mit dieser Diplomarbeit werden Schema- und Datenmigrationen von einem kommerziellen Datenbankmanagementsystem auf das eines anderen Herstellers durchgeführt und dargelegt, welche Probleme bei einer Umstellung bewältigt werden müssen.

Die Datenbankmigrationen konnten sowohl von SQL Server zu Oracle als auch von Oracle zu SQL Server durchgeführt werden. Es hat sich aber herausgestellt, dass es noch nicht möglich ist, mit Hilfe von Migrationswerkzeugen automatisch T-SQL Code in PL/SQL Code oder umgekehrt fehlerfrei zu konvertieren. Eine manuelle Nachbearbeitung ist notwendig. Weiterhin kann auch größtenteils kein Native PL/SQL bzw. Native T-SQL erstellt werden, weil viele Schlüsselwörter nur anhand von eigens dafür angelegten Paketen oder sogar nur mittels einer eigens dafür angelegten Datenbank konvertiert werden konnten. Beispiele haben gezeigt, dass ein Migrationswerkzeug nicht die Logik hinter einem Programm-Code verstehen kann und somit beispielsweise nicht immer in den idealen Datentyp konvertiert. Bei der Migration von Oracle zu SQL Server hat sich herausgestellt, dass sich der ursprüngliche PL/SQL Code verdreifacht hat, weil für die Konvertierung der PL/SQL Prozeduren T-SQL Prozeduren und Wrapper-Funktion, die die implementierten T-SQL Prozeduren aufrufen, erzeugt wurden. Dies alles führte dazu, dass der erzeugte Code eine sehr schlechte Performanz aufwies.

Bei einem Vergleich der Architekturen, der Datenbankobjekte und der Sperrkonzepte wurde deutlich, dass auf beiden Seiten einige Hersteller-Dokumentationen über fremde Datenbankmanagementsysteme fehlerhaft und unvollständig sind. Deswegen sollten vor einer Migration immer die ursprünglichen Hersteller-Dokumentationen betrachtet werden.

Obwohl beide Datenbankmanagementsysteme den ANSI-Standard unterstützen, wurde erkennbar, dass sich beispielsweise die in der Praxis verwendete Oracle-Syntax in entscheidenden Punkten von dem ANSI-Standard unterscheidet.

Bei der Untersuchung der Sperrkonzepte hat sich ergeben dass beispielsweise bei einer automatischen Konvertierung logischer SQL Server-Transaktionen in logische Oracle-Transaktionen die vorher in SQL Server seriell ausgeführten Transaktionen in Oracle einen Deadlock verursachen können.

Das Ergebnis dieser Diplomarbeit zeigt, dass sogar eine Migration einer einfachen Beispieldatenbank mit einem nicht geringem technischen Aufwand verbunden ist, da durch eine automatische Konvertierung keine durchweg brauchbaren Ergebnisse erzielt werden können. Daher sollten Unternehmen vor einem Wechsel von einem kommerziellen Datenbankmanagementsystem auf das eines anderen Herstellers auch die versteckten Personalkosten einkalkulieren. Dies betrifft besonders den Aufwand den ursprünglichen Code in einen syntaktischen, semantischen und performanten Code zu konvertieren.

Literaturverzeichnis

[Ahrends06] Ahrends, Johannes; Lenz, Dierk; Schwanke, Patrick: Oracle 10g für den DBA – Effizient konfigurieren, optimieren und verwalten, 1. Aufl., München: Addison-Wesley, 2006.

[Best05] Best, Tom; Billings, M.J.: Oracle Database 10g: Administration Workshop I – Schulungsunterlagen, Band 1, Oracle, 2005.

[Dröge06] Dröge, Ruprecht; Raatz, Markus: SQL Server 2005 – Konfigurierung, Administration, Programmierung, 2. Aufl., Unterschleißheim: Microsoft Press Deutschland, 2006.

[FaeskScript] Faeskorn-Woyke, Heide; Datenbanken und Informationssysteme Teil 1, Datenbanksicherheit und Transaktionen, Land Nordrhein-Westfalen: Ministerium für Schule, Wissenschaft und Forschung 2001.

[Konop07] Konopasek, Klemens; Tiemeyer, Ernst: SQL Server 2005 – Der schnelle Einstieg, Abfragen, Transact-SQL, Entwicklung und Verwaltung, 1. Aufl., München: Addison-Wesley, 2007.

[MA_HLP] o.V., o.Datum. "Converting Oracle Schemas" <SSMA-Programmhilfe: Migrating Oracle Database to SQL Server/ Converting Oracle Schemas> (30.07.2007)

[MA_HLP2] o.V., o.Datum. "Project Settings (Conversion)" <SSMA-Programmhilfe: User Interface Reference/Project Reference (Settings)> (30.07.2007)

[MSDN1] o.V., 05.12.2005. “Physische Datenbankdateien und Dateigruppen“ <http://msdn2.microsoft.com/de-de/library/ ms179316.aspx> (27.04.2007)

[MSDN2] o.V., o. Datum. “Datentypzuordnung für Oracle-Verleger“ <http://msdn2.microsoft.com/de-de/library/ms151817.aspx> (08.05.2007)

[MSDN3] o.V., 17.07.2006. “Reservierte Schlüsselwörter (Transact-SQL)“ <http://msdn2.microsoft.com/de-de/library/ms189822.aspx> (14.05.2007)

[MSDN4] o.V., o. Datum. “Überlegungen zum Entwurf und Einschränkungen für Oracle-Verleger“<http://msdn2.microsoft. com/de-de/library/ms151859.aspx> (10.05.2007)

[MSDN5] o.V., 01.02.2007. “Spezifikationen der maximalen Kapazität für SQL Server 2005“ <http://msdn2 .microsoft.com/de-de/library/ ms143432.aspx> (10.05.2007)

[MSOD1] o.V., 04.2007. “Systemdatenbanken“ <ms-elp://MS.SQLCC.v9/ MS.SQLSVR.v9.de/udb9/ html/30468a7c-4225-4d35-aa4a-ffa7da4f1282.htm> (02.05.2007)

[MSOD2] o.V., 02.2007. “Grundlegendes zu Dateien und Dateigruppen“ <ms-help://MS.SQLCC.v9/MS.SQLSVR.v9.de/udb9/html/ 9ca11918-480d-4838-9198-cec221ef6ad0.htm> (27.04.2007)

[MSOD3] o.V., 02.2007. “Instanzname“ <ms-help://MS.SQLCC.v9/ MS.SQLSVR.v9.de/instsql9/html/5bf822fc-6dec-4806-a153-e200af28e9a5.htm> (05.05.2007)

[MSOD4] o.V., 02.2007. “Sperrmodi” <ms-help://MS.SQLCC.v9/ MS. SQLSVR. 9.de/udb9/html/ 108297fa-35fc-4cbe-a1ac-369aabd8cf44.htm> (23.08.2007)

[MSOD5] o.V., 02.2007. “Isolationsstufen im Datenbankmodul” <ms-help://MS.SQLCC.v9/MS.SQLSVR.v9.de/udb9/html/ 8ac7780b-5147-420b-a539-4eb556e908a7.htm> (23.08.2007)

[MSTN1] o.V., o. Datum. “Chapter 7 - Migrating Oracle Databases to SQL Server 2000” <http://www.microsoft.com/technet/ prodtechnol /sql/2000/reskit/part2/c0761.mspx?mfr=true> (10.06.2007)

[MurDOC] Murray, Chuck, 06.2007 “Oracle® Database SQL Developer Supplementary Information for Microsoft SQL Server Migrations Release 1.2 <http://download.oracle.com/docs/cd/ E10405_01/doc/appdev.120/e10379/ss_oracle_compared.htm #sthref21> (20.08.2007)

[OraDOC1] o.V., o. Datum. “B Oracle Reserved Words, Keywords, and Namespaces”<http://download-uk.oracle.com/docs/cd/ B19306_01/appdev.102/b14354/appb.htm> (16.05.2007)

[OraDOC2] o.V., o. Datum. “D PL/SQL Reserved Words and Keywords” <http://download-uk.oracle.com/docs/cd/B19306_01/appdev. 102/b14261/reservewords.htm> (16.05.2007)

[OraDEV] o.V., 05.08.2007. “SQL Developer-Download: SQL Server Migration Assistant for Oracle V3.0” <http://www.microsoft. com/downloads/details.aspx? FamilyId=0E06A04C-D0C3-4F31 -B201-FBE7C25F32FB&displaylang=en> (10.08.2007)

[OraFRM] o.V., 08.08.2007.“Thread: *URGENT* Does Oracle SQL Developer able to migrate tables relationships?” <http://forums. oracle.com/forums/thread.jspa?messageID=1919994&#1919994 (14.08.2007)

[OraMWB] o.V., 06.2005. “Oracle® Migration Workbench User’s Guide Release 10.1.0.4 for Microsoft Windows 98/2000/NT/XP and Linux x86“ <http://www.oracle.com/technology/tech/migration/workbench/ htdocs/101040/user_guide/book.pdf> (13.08.2007)

[SSMA] o.V., o.Datum. “SSMA-Download: SQL Server Migration Assistant for Oracle V3.0” <http://www.microsoft.com/downloads/details.aspx?familyid=0E06A04C-D0C3-4F31-B201-FBE7C25F32FB&displaylang=en> (25.07.2007)

[SSMA1] o.V., o. Datum. “Häufig gestellte Fragen zum SQL Server Migration Assistant“ <http://www.microsoft.com/germany/sql/migration/ssma_faq.mspx> (25.07.2007)

[ToadData] o.V., o. Datum “Toad Data Modeler” <http://www.quest.com/Toad-Data-Modeler/> (03.06.2007)

[ToadOra] o.V., o. Datum “Toad for Oracle“ <http://www.quest.com/toad-for-oracle/> (03.06.2007)

[WAWI] Konopasek, Klemens; Tiemeyer, Ernst: Buch-CD SQL Server 2005 – Der schnelle Einstieg, Abfragen, Transact-SQL, Entwicklung und Verwaltung, 1. Aufl., München: Addison-Wesley, 2007

[Wilde04] Wilde, Antje, 23.05.2004. “PL/SQL: Eine Einführung”<http://www.a-wilde.de/hp/studium/db/plsql4.htm> (13.05.2007)

Anhang A

Maximale Kapazität für SQL Server 2005

SQL Server 2005-Datenbankmodul-Objekt	Maximale Größe/Anzahl SQL Server 2005
Batchgröße[1]	65.536 * Netzwerkpaketgröße
Bytes pro Spalte mit kurzen Zeichenfolgen	8,000
Bytes pro GROUP BY, ORDER BY	8,060
Bytes pro Indexschlüssel[2]	900
Bytes pro Fremdschlüssel	900
Bytes pro Primärschlüssel	900
Bytes pro Zeile[8]	8,060
Bytes im Quelltext einer gespeicherten Prozedur	Batchgröße oder 250 MB, je nachdem, welcher Wert niedriger ist
Bypes pro varchar(max)-, varbinary(max)-, xml-, text- oder image-Spalte.	2^{31-1}
Zeichen pro ntext- oder nvarchar(max)-Spalte	2^{30-1}
Gruppierte Indizes pro Tabelle	1
Spalten in GROUP BY, ORDER BY	Begrenzung nur durch die Anzahl von Bytes
Spalten oder Ausdrücke in einer GROUP BY WITH CUBE- oder WITH ROLLUP-Anweisung	10
Spalten pro Indexschlüssel[7]	16
Spalten pro Fremdschlüssel	16
Spalten pro Primärschlüssel	16
Spalten pro Basistabelle	1,024
Spalten pro SELECT-Anweisung	4,096
Spalten pro INSERT-Anweisung	1,024

Verbindungen pro Client	Höchstwert konfigurierter Verbindungen
Datenbankgröße	1.048.516 Terabytes
Datenbanken pro Instanz von SQL Server	32,767
Dateigruppen pro Datenbank	32,767
Dateien pro Datenbank	32,767
Dateigröße (Daten)	16 Terabytes
Dateigröße (Protokoll)	2 Terabytes
Verweise auf Fremdschlüsseltabellen pro Tabelle[4]	253
Bezeichnerlänge (in Zeichen)	128
Instanzen pro Computer	50 Instanzen auf einem eigenständigen Server 25 Instanzen auf einem Failovercluster
Länge einer Zeichenfolge, die SQL-Anweisungen enthält (Batchgröße)[1]	65.536 * Netzwerkpaketgröße
Sperren pro Verbindung	Maximale Anzahl Sperren pro Server
Sperren pro Instanz von SQL Server[5]	32-Bit-Version: Bis zu 2.147.483.647 64-Bit-Version: Begrenzung nur durch Arbeitsspeicher
Schachtelungsebenen gespeicherter Prozeduren[6]	32
Geschachtelte Unterabfragen	32
Schachtelungsebenen für Trigger	32
Nicht gruppierte Indizes pro Tabelle	249
Parameter pro gespeicherter Prozedur	2,100
Parameter pro benutzerdefinierter Funktion	2,100
REFERENCES pro Tabelle	253

Zeilen pro Tabelle	Begrenzung durch verfügbaren Speicherplatz
Tabellen pro Datenbank[3]	Begrenzung durch die Anzahl der Objekte in einer Datenbank
Partitionen pro partitionierter Tabelle oder partitioniertem Index	1,000
Statistiken für nicht indizierte Spalten	2,000
Tabellen pro SELECT-Anweisung	256
Trigger pro Tabelle[3]	Begrenzung durch die Anzahl der Objekte in einer Datenbank
UNIQUE-Indizes oder -Einschränkungen pro Tabelle	249 nicht gruppierte und 1 gruppierter
Benutzerverbindungen	32,767
XML-Indizes	249

Tabelle 21: Maximale Kapazität für SQL Server 2005

[1] Die Netzwerk-Paketgröße entspricht der Größe der TDS-Pakete (Tabular Data Stream), die für die Kommunikation zwischen Anwendungen und relationalem Datenbankmodul verwendet werden. Die Standardpaketgröße beträgt 4 Kilobytes (KB).

[2] Die maximale Anzahl von Bytes in einem beliebigen Indexschlüssel kann den Wert 900 in SQL Server 2005 nicht überschreiten. Man kann einen Schlüssel mit Hilfe von Spalten variabler Länge definieren, deren maximale Größen zusammen mehr als 900 Bytes betragen, wenn niemals eine Zeile eingefügt wird, die in diesen Spalten mehr als 900 Bytes an Daten enthält. In SQL Server 2005 können Nichtschlüsselspalten in den nicht gruppierten Index aufgenommen werden, um die maximale Indexschlüsselgröße von 900 Bytes zu vermeiden.

[3] Zu den Datenbankobjekten zählen Tabellen, Sichten, gespeicherte Prozeduren, benutzerdefinierte Funktionen, Trigger, Regeln, Standardwerte und Einschränkungen. Die Summe aller Objekte in einer Datenbank kann 2.147.483.647 nicht übersteigen.

[4] Auch wenn eine Tabelle eine unbeschränkte Anzahl von FOREIGN KEY-Beschränkungen enthalten kann, beträgt das empfohlene Maximum 253. In Abhängigkeit von der Hardwarekonfiguration, die SQL Server hostet, kann das Angeben weiterer Fremdschlüssel den Abfrageoptimierer bei der Verarbeitung stark beanspruchen.

[5] Dieser Wert dient der statischen Sperrenzuordnung. Dynamische Sperren sind nur durch den Arbeitsspeicher beschränkt.

[6] Wenn eine gespeicherte Prozedur auf mehr als 8 Datenbanken zugreift oder sich mehr als 2 Datenbanken überlappen, erhält man einen Fehler.

[7] Wenn die Tabelle einen oder mehrere XML-Indizes enthält, ist der Gruppierungsschlüssel der Benutzertabelle auf 15 Spalten beschränkt, das die XML-Spalte dem Gruppierungsschlüssel des primären XML-Index hinzugefügt wird. In SQL Server 2005 können Nichtschlüsselspalten in den nicht gruppierten Index aufgenommen werden, um die Beschränkung auf maximal 16 Schlüsselspalten zu vermeiden.

[8] SQL Server 2005 unterstützt die Zeilenüberlaufspeicherung, sodass Spalten variabler Länge aus der Zeile verschoben werden können. Für Spalten variabler Länge, die aus der Zeile verschoben wurden, wird im Hauptdatensatz nur ein 24-Byte-Stamm gespeichert. Aus diesem Grund ist das tatsächlich gültige Zeilenlimit höher als in früheren Versionen von SQL Server.

Anhang B

Inhalt der beigefügten CD-ROM

Die beigefügte CD-ROM enthält folgende ergänzende Unterlagen zur Diplomarbeit:

- Quelltext der SQL-Dateien zur Erstellung der Datenbankstrukturen und der Füllung der Datenbank-Tabellen.

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zur haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift E. Türkeri