Darfs auch mal was anderes sein? – PL/SQL und T-SQL unter der Vergleichslupe!

Anja Zahn

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

2013 © Trivadis

trivadis
makes IT easier.

AGENDA

1. Fallstricke, oft gebrauchtes
2. Error Handling
3. Transaction handling inkl. Logging
4. Trigger
5. Set-based Operationen/ OUTPUT-Clause

2013 © Trivadis

trivadis
makes IT easier.

Fallstricke, oft gebrauchtes

2013 © Trivadis

trivadis
makes IT easier.

Grundlegendes SQL Server <> Oracle

- eine Oracle-Instanz → eine DB
- eine SQLServer Instanz → mehrere DB
- In Oracle: User = Schema (sobald der User Objekte hat!)
- In SQLServer werden User explizit verwaltet, es gibt keine direkte Zusammengehörigkeit User ←→ Schema
- Data Dictionary = INFORMATION_SCHEMA in der Systemdatenbank master
  - hier lassen sich alle relevanten Daten auslesen (Tabellen, Spalten, Constraints etc.pp)
- Objekte-Namen
  - Oracle: bis zu 30 Zeichen
  - SQLServer: bis zu 128 Zeichen

2013 © Trivadis

trivadis makes IT easier.

Datentypen

Oracle		SQLServer
VARCHAR2(n)	=	VARCHAR(n)
TIMESTAMP	!=	TIMESTAMP (automatisch, nur 1 pro tabelle)
FLOAT	=	FLOAT, REAL etc
NUMBER(N)	=	INTEGER, SMALLINT, TINYINT
DATE	=	DATETIME (nicht DATE!)
CLOB	=	TEXT
BLOB	=	IMAGE

2013 © Trivadis

trivadis makes IT easier.

Syntaxunterschiede

Oracle	SQLServer
CONNECT user_name/password	USE database_name
‘a‘ || ‘b‘	‘a‘+‘b‘
DBMS_LOCK.sleep	WAITFOR
Substr	Substring
instr	Charindex
nvl	isnull
SYSDATE	GETDATE()
SELECT ‚Hello World‘ from dual (select ohne from nicht möglich)	Select ‚Hello World‘
to_char, to_number, to_date, chartorowid, rowtochar, hextochar, chartohex	convert(data type, expression, [format])
DESCRIBE	sp_help

2013 © Trivadis

trivadis makes IT easier.

Syntaxunterschiede

	Oracle	SQLServer
Select INTO - Statement	INSERT into target_table SELECT col1, col2, col3 FROM source_table WHERE where_clause	SELECT col1, col2, col3 INTO target_table FROM source_table WHERE where_clause → Oracle-syntax auch möglich
Subqueries anstatt Spalten	SELECT year, DECODE(qtr, 1, amt, 0) q1, DECODE(qtr, 2, amt, 0) q2, DECODE(qtr, 3, amt, 0) q3, DECODE(qtr, 4, amt, 0) q4 FROM sales s;	SELECT year, CHOOSE(qtr, 1, amt, 0) q1, CHOOSE(qtr, 2, amt, 0) q2, CHOOSE(qtr, 3, amt, 0) q3, CHOOSE(qtr, 4, amt, 0) q4 FROM sales s;
Insert into	nur auf view mit einer unterliegenden Tabelle	In View auf mehrere Tabellen möglich ---> nur eine Tabelle darf betroffen sein
Update/Delete from (basierend auf Werten aus anderen Tabellen)	Nicht möglich	update titles SET pub_id = publishers.pub_id FROM titles, publishers WHERE titles.title LIKE 'C%' AND publishers.pub_name = 'new age'

2013 © Trivadis

trivadis makes IT easier.

Allgemeines

	Oracle	SQLServer
Joins	Where col1 = col2 (+)	na
Automatisches Füllen des PK	Sequence + Trigger	Identity-Column (spalte bigint IDENTITY(1,1) NOT NULL)
Packages	Hier Anlegen/Nutzen globaler Variablen möglich	Keine → somit auch keine globalen Variablen (muss über temp. Tabellen gelöst werden)
Procedures	Kein Select ohne Into	Ausgabe eines Selects möglich → gut zum debuggen
Indexes	Wird bei PK, FK und UK automatisch generiert	Nur bei PK und UK automatisch
Functions	Können dml	Können kein dml
Procedures	Kein direktes DDL (über dynamisches SQL)	DDL direkt ausführbar
NULL	NULL = NULL → FALSE	NULL = NULL → TRUE (bei Standardeinstellung ANSI_NULLS OFF)
String literal	'x'	'x' und "x"
Null in strings	Variable = NULL → der restliche Ausdruck wird ausgewertet	Variable = NULL → der restliche Ausdruck wird nicht ausgewertet (wird null gesetzt)
Date-Functions (komplett verschieden)	Siehe Referenz	Siehe Referenz

2013 © Trivadis

trivadis makes IT easier.

Procedurales I

	Oracle	SQLServer

Aufbau prozeduraler Block	DECLARE
	BEGIN
	END;
		BEGIN
		DECLARE
		END

Lokale Variable in DECLARE initialisieren	DECLARE var_name type := value;	DECLARE @var_name type = value;
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>
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

2013 © Trivadis

trivadis makes IT easier.

Procedurales II

	Oracle	SQLServer

If-Anweisung	IF ... THEN
	ELSIF ... THEN
	ELSE
	ENDIF;
	Alternativ:
	CASE WHEN ... THEN...;
	ELSE ... ;
	END CASE;
		IF ... [BEGIN ... END]
		ELSE ... [BEGIN ... END]
		ELSE IF ...
		
		Alternativ:
		CASE WHEN ... THEN...;
		ELSE ... ;
		END CASE;

While-Schleife	WHILE ... LOOP END LOOP;	WHILE ... [BEGIN ... END]
Schleifen-kontrolle	FOR ... END LOOP; LOOP ... END LOOP;	<nicht unterstützt>
Schleifenabbruch	EXIT, EXIT WHEN	BREAK
Print output	DBMS_OUTPUT.PUT_LINE	PRINT
Raise error (benutzerdefiniert)	RAISE_APPLICATION_ERROR	RAISERROR
Rückgabe aus Prozeduren	Out-Parameter	RETURN (nur int)
Collection	TYPE type_name IS TABLE OF element_type [NOT NULL];	<nicht unterstützt>

2013 © Trivadis

trivadis makes IT easier.

Error Handling

2013 © Trivadis

trivadis
makes IT easier.

Exceptionhandling T-SQL

- SYNTAX

```
BEGIN TRY
  // some Statement here
END TRY
BEGIN CATCH
 // some Statement here;
END CATCH;
```

- Auf einen TRY-Block muss immer ein CATCH-Block folgen
- Es werden nur Fehler mit einem Fehlergrad von 11-19 vom CATCH-Block verarbeitet.
  - <=10 → nur Informationen
  - >=20 → Verbindung zur Datenbank wird abgebrochen
- Können ineinander verschachtelt werden

2013 © Trivadis

trivadis makes IT easier.

Exceptionhandling T-SQL

Beispiel Ausgabe Error-Informationen

```
BEGIN TRANSACTION
BEGIN TRY
    // some SQL statements here

    COMMIT TRANSACTION
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() AS ErrorState,
        ERROR_PROCEDURE() AS ErrorProcedure,
        ERROR_LINE() AS ErrorLine,
        ERROR_MESSAGE() AS ErrorMessage

    ROLLBACK TRANSACTION
END CATCH
```


2013 © Trivadis

trivadis makes IT easier.

Exceptionhandling PL/SQL

- SYNTAX

```
DECLARE
  Declaration section
BEGIN
  Exception section
EXCEPTION
WHEN ex_name1 THEN
   -Error handling statements
WHEN ex_name2 THEN
   -Error handling statements
WHEN Others THEN
  -Error handling statements
END;
```

- Können ineinander verschachtelt werden
- Abarbeitung von oben nach unten
- Informationen über SQLCODE and SQLERRM

2013 © Trivadis

trivadis makes IT easier.

User defined error – T-SQL

RAISEERROR → Benutzerdefinierte Exceptions

- Kann im TRY- und im CATCH-Block ausgeführt werden
- Bei Schweregrad 11-19
  - TRY: zugehöriger CATCH-Block wird aufgerufen
  - CATCH: aufrufende Anwendung bzw. Batch wird aufgerufen
- Fehlernummer für RAISERROR muss >= 50000 sein

2013 © Trivadis

trivadis makes IT easier.

User defined error – PL/SQL

- Eigene Exception definieren + RAISE_APPLICATION_ERROR (error_number, error_message);
- Errornummer -20000 bis -20999

```
 DECLARE
  my_exception EXCEPTION;

BEGIN
  Raise my_exception ;
EXCEPTION
   WHEN my_exception
   THEN
        raise_application_error(-20001, Hier kommt meine
Exception.');
END;

/
```


2013 © Trivadis

trivadis makes IT easier.

Exception “werfen“

- PL/SQL: RAISE;
- T-SQL: THROW
  - Erst ab SQLServer 2012
- Bei beiden wird die aktuelle Errormeldung an den nächsten Exception-Block bzw. die aufrufende Einheit weitergegeben

2013 © Trivadis

trivadis
makes IT easier.

Transaction handling inkl. Logging

2013 © Trivadis

trivadis
makes IT easier.

Transaction handling PL/SQL

- Transaktionen beginnen implizit mit dem ersten ausführbaren Statement

und enden mit:

  - Einem commit; scheibt alle schwebenden Änderungen in der DB fest
  - Einem rollback; rollt alle schwebenden Änderungen zurück

- Vorsicht bei implizitem Commit:
  - DDL Statement
  - Nutzer loggt sich aus

2013 © Trivadis

trivadis makes IT easier.

Logging PL/SQL

Innerhalb eines PL/SQL-Block können Daten „zwischengespeichert“ werden

```
Declare
Begin
Update table xy
Commit;

Insert into table xy...
Rollback;
End;
```

- Wenn nur ein Teil der Daten festgeschrieben werden soll
  - Autonome Transacktionen

```
CREATE PROCEDURE abc AS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
COMMIT;
END abc;
/
```


2013 © Trivadis

trivadis makes IT easier.

Transaction handling T-SQL

- Default Transaktionsmodus ist „Autocommit“
  - Jede Anweisung erfährt automatisch einen commit oder rollback
- Vergleichbar mit Oracle ist der Modus IMPLICIT_TRANSACTIONS
  - Nach jedem commit/rollback wird mit der nächsten Anweisung implizit eine Transaktion geöffnet
  - Transaktionen können explizit eröffnet und verschachtelt werden
    - Mit jeder geöffneten Transaktion erhöht sich @@TRANCOUNT um 1
  - Ein **commit**
    - Setzt @@TRANCOUNT um 1 herunter
    - Bei @@TRANCOUNT= 1 → scheibt alle schwebenden Änderungen in der DB fest
  - Ein **rollback**:
    - setzt @@TRANCOUNT auf 0 und rollt alle schwebenden Änderungen zurück

2013 © Trivadis

trivadis makes IT easier.

Logging T-SQL

- In SQLServer gibt es keine autonomen Transaktionen
- Es gibt keine Möglichkeit Informationen wegzuschreiben, wenn die Transaktion nachfolgend zurück gerollt wird
  - Dabei werden Einträge in Log-Tabellen ebenfalls entfernt
- Workaround:
  - Loopback Linked Server
    - Linked Server entspricht einem Database-Link in Oracle
    - Loopback heißt, dass der gleiche Server angesprochen wird
    - Es wird quasi eine neue Session aufgebaut
  - 2. Option sind die Table Variablen
    - Diese sind nicht von dem Ende einer Transaktion betroffen
    - Auch wenn eine Transaktion zurück gerollt wird, sind die Daten in der Table Variablen noch verfügbar und müssen dann weggeschrieben werden
  - Vorteil Loopback: eine parallele Session sieht die Änderungen bereits

2013 © Trivadis

trivadis makes IT easier.

Logging T-SQL

Anlegen eines Loopback Linked Server:

```
USE MASTER
GO

EXEC sp_addlinkedserver @server = N'loopback',@srvproduct = N'
',@provider = N'SQLNCLI', @datasrc = @@SERVERNAME
GO
EXEC sp_serveroption loopback,N'remote proc transaction
promotion','FALSE'
Go
```

Verwendung:

```
BEGIN TRAN InnerTran

   EXEC loopback.tempdb.dbo.usp_ErrorLogging @ERROR

COMMIT TRAN InnerTran
```


2013 © Trivadis

trivadis makes IT easier.

Logging T-SQL

Verwendung der Tablevariablen:

```
DECLARE @errorlog TABLE
(logTime DATETIME, msg VARCHAR(255));

declare  @errNumber int = 50001

BEGIN TRANSACTION

  INSERT INTO @errorlog (logTime , msg )
  VALUES (GETDATE(), 'Error ' + CAST(@errNumber AS VARCHAR(8)) +'
       occurred.')

ROLLBACK TRANSACTION

SELECT * FROM @errorlog
```

Logging.sql.txt

2013 © Trivadis

trivadis
makes IT easier.

Trigger

2013 © Trivadis

trivadis
makes IT easier.

Triggerarten PL/SQL

- DDL-Trigger
- Database-Trigger (SERVERERROR, LOGON, LOGOFF, STARTUP, SHUTDOWN)
- DML-Trigger
  - Auf Tabellen:
    - Before/After Statement
    - Before/After each row
      - Beziehen sich auf das auslösende Statement
  - Auf Views
    - INSTEAD OF-Trigger → werden anstatt des auslösenden Statements ausgeführt
  - Zugriff auf die Daten vor und nach dem Statement über OLD /NEW

2013 © Trivadis

trivadis makes IT easier.

Triggerarten T-SQL

- DDL-Trigger
- LOGON-Trigger
- DML-Trigger
  - Deleted/inserted entspricht OLD /NEW bei ORACLE
  - Keine ROW-Level- Trigger, alle beziehen sich auf das Statement
  - Keine BEFORE-Trigger
  - AFTER-Trigger nur auf Tabellen
    - Nur nach erfolgreichem Abschluss des INSERT/UPDATE/DELETE
    - Spalten vom Typ text, ntext und image können nicht verarbeitet werden → instead of notwendig
  - INSTEAD OF-Trigger:
    - nur einer pro Tabelle/View pro INSERT/UPDATE/DELETE
    - werden anstatt des auslösenden Statements ausgeführt

2013 © Trivadis

trivadis makes IT easier.

Set-based Operationen/ OUTPUT-Clause

2013 © Trivadis

trivadis makes IT easier.

Set-based Operationen

- SELECT INTO from
- UPDATE / DELETE from
- EXCEPT / INTERSECT / UNION / UNION ALL
- MERGE
- Helfer:
  - Table Variable
  - #Table

2013 © Trivadis

trivadis makes IT easier.

Cursor vs. Set-based (Update from)

```
--Cursor
    DECLARE @person_id INT
    DECLARE @field2 VARCHAR (10)

    DECLARE TestCursor3 CURSOR FOR
       SELECT t2.person_id, t2.field2 FROM table2 t2

    OPEN TestCursor3

    FETCH NEXT FROM TestCursor3
    INTO @person_id, @field2

    WHILE @@FETCH_STATUS = 0
    BEGIN
       UPDATE table1 SET field2 = @field2
       WHERE person_id =  @person_id

       FETCH NEXT FROM TestCursor3 INTO  @person_id, @field2
    END

    CLOSE TestCursor3
    DEALLOCATE TestCursor3

--Set
 UPDATE t1 SET field2 = t2.field2
    --SELECT t1.field2, t2, field2, t1.person_id  -- only for debugging
    FROM table1 t1
    JOIN table2 t2 ON t2.person_id = t1.person_id
```


2013 © Trivadis

trivadis makes IT easier.

OUTPUT-Clause - INSERT

```
USE AdventureWorks2012;
GO
IF OBJECT_ID('tempdb..#MyTable') IS NOT NULL
DROP TABLE dbo.#MyTable
Create table #MyTable (
    NewScrapReasonID smallint
   ,Name varchar(50)
   ,ModifiedDate datetime
);
INSERT Production.ScrapReason
OUTPUT INSERTED.ScrapReasonID, INSERTED.Name,INSERTED.ModifiedDate
INTO #MyTable
VALUES (N'Operator error', GETDATE());

--Display the result set of the table variable.
SELECT NewScrapReasonID, Name, ModifiedDate
FROM #MyTable;

--Display the result set of the table.
SELECT ScrapReasonID, Name, ModifiedDate
FROM Production.ScrapReason;
GO
```

OUTPUT_CLAUSE.txt

2013 © Trivadis

trivadis
makes IT easier.

OUTPUT-Clause - UPDATE

```
USE AdventureWorks2012;
GO
DECLARE @MyTableVar table(
    EmpID int NOT NULL,
    OldVacationHours int,
    NewVacationHours int,
    ModifiedDate datetime);

UPDATE TOP (10) HumanResources.Employee
SET VacationHours = VacationHours * 1.25,  ModifiedDate = GETDATE()
OUTPUT inserted.BusinessEntityID,
       deleted.VacationHours,
       inserted.VacationHours,
       inserted.ModifiedDate
INTO @MyTableVar;

--Display the result set of the table variable.
SELECT EmpID, OldVacationHours, NewVacationHours, ModifiedDate
FROM @MyTableVar;
GO

--Display the result set of the table.
SELECT TOP (10) BusinessEntityID, VacationHours, ModifiedDate
FROM HumanResources.Employee;
GO
```


2013 © Trivadis

trivadis makes IT easier.

OUTPUT-Clause - DELETE

```
USE AdventureWorks2012;
GO

DELETE Sales.ShoppingCartItem
OUTPUT DELETED.*
WHERE ShoppingCartID = 20621;

--Verify the rows in the table matching the WHERE clause have been
deleted.

SELECT COUNT(*) AS [Rows in Table] FROM Sales.ShoppingCartItem
WHERE ShoppingCartID = 20621;
GO
```


2013 © Trivadis

trivadis makes IT easier.

Table Functions

2013 © Trivadis

trivadis
makes IT easier.

Table Valued Functions PL/SQL

```
CREATE TYPE AType AS OBJECT (Department VARCHAR(64), Employee VARCHAR(64));
/
CREATE TYPE ATypeCol AS TABLE OF AType;
/
CREATE OR REPLACE FUNCTION fEmployee ( dep IN VARCHAR )
RETURN ATypeCol PIPELINED IS
BEGIN
  FOR i IN (SELECT Department.Name, Employee.FirstName || ' ' || Employee.LastName Employee
            FROM Employee JOIN Department ON DepartmentId = Department.Id
            WHERE Department.Name = dep) LOOP
            PIPE ROW(AType(i.Name, i.Employee));
  END LOOP;
  RETURN;
END;
/
SELECT * FROM TABLE(fEmployee('IT'));
/
```


2013 © Trivadis

trivadis makes IT easier.

Table Valued Functions T-SQL

```
CREATE FUNCTION Sales.ufn_SalesByStore (@storeid int)
RETURNS TABLE
AS
RETURN
(
    SELECT P.ProductID, P.Name, SUM(SD.LineTotal) AS 'Total'
    FROM Production.Product AS P
    JOIN Sales.SalesOrderDetail AS SD ON SD.ProductID = P.ProductID
    JOIN Sales.SalesOrderHeader AS SH ON SH.SalesOrderID = SD.SalesOrderID
    JOIN Sales.Customer AS C ON SH.CustomerID = C.CustomerID
    WHERE C.StoreID = @storeid
    GROUP BY P.ProductID, P.Name
);


SELECT * FROM Sales.ufn_SalesByStore (123)
```


2013 © Trivadis

trivadis
makes IT easier.

Fragen und Antworten...

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

2013 © Trivadis

05.04.2015

trivadis
makes IT easier.

Links

Rekursive CTE: http://technet.microsoft.com/de-de/library/ms186243%28v=sql.105%29.aspx

Loopback Linked Server:http://blogs.msdn.com/b/sqlprogrammability/archive/2008/08/22/how-to-create-an-autonomous-transaction-in-sql-server-2008.aspx

Transaktionsverhalten
https://technet.microsoft.com/de-de/library/ms190612%28v=sql.105%29.aspx

http://dbs.informatik.uni-halle.de/Lehre/PrakDB_SS08/8_SQL_trans.pdf

Trigger:

http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_7004.htm

https://msdn.microsoft.com/de-de/library/ms189799.aspx

2013 © Trivadis

trivadis makes IT easier.

Isolation Level und Locking Links

- SET TRANSACTION ISOLATION LEVEL
  - https://technet.microsoft.com/de-de/library/ms173763(v=sql.110).aspx
- BOL Concurrency effects
  - https://technet.microsoft.com/en-us/library/ms190805(v=sql.105).aspx
- Understanding SQL Server Locking
  - http://www.mssqltips.com/sqlservertip/1968/understanding-sql-server-locking/
- Introduction to Locking in SQL Server
  - http://www.sqlteam.com/article/introduction-to-locking-in-sql-server
- Demonstrations of Transaction Isolation Levels in SQL Server
  - http://www.mssqltips.com/sqlservertip/2977/demonstrations-of-transaction-isolation-levels-in-sql-server/

2013 © Trivadis

trivadis makes IT easier.

Community Links

Community Oracle:

https://community.oracle.com/welcome

http://sql-plsql-de.blogspot.de/

Community SQL Server:

http://www.sqlservercentral.com/

2013 © Trivadis

trivadis
makes IT easier.

Security Links

- Security and Protection (Database Engine)
  - https://msdn.microsoft.com/en-us/library/bb510589(v=sql.110).aspx
- SQL Server Best Practices – Implementation of Database Object Schemas
  - https://msdn.microsoft.com/en-us/library/dd283095.aspx
- SQL Server Security Blog
  - http://blogs.msdn.com/b/sqlsecurity/
- Security Tips
  - http://www.mssqltips.com/sql-server-tip-category/19/security/
- Engine Separation of Duties for the Application Developer
  - https://msdn.microsoft.com/en-us/library/cc974525(SQL.100).aspx

2013 © Trivadis

trivadis makes IT easier.

Index and Statistic Links

- Indexes
  - https://msdn.microsoft.com/en-us/library/ms175049(v=sql.110).aspx
- Columnstore Indexes
  - https://msdn.microsoft.com/en-us/library/gg492088(v=SQL.110).aspx
- SQL Server Columnstore Index FAQ
  - http://social.technet.microsoft.com/wiki/contents/articles/3540.sql-server-columnstore-index-faq.aspx
- Statistic
  - https://msdn.microsoft.com/en-us/library/ms190397(v=sql.110).aspx

2013 © Trivadis

trivadis makes IT easier.

Execution Plan and other Performance-Links

- Checklist: SQL Server Performance
  - https://msdn.microsoft.com/en-us/library/ff647681.aspx
- SQL Server Index Design Guide
  - https://technet.microsoft.com/en-us/library/jj835095(v=sql.110).aspx
- Monitor and Tune for Performance
  - https://msdn.microsoft.com/en-us/library/ms189081(v=sql.110).aspx

2013 © Trivadis

trivadis makes IT easier.