Wie Oracle-kompatibel ist DB2 oder ist DB2 das bessere Oracle?

Uwe Simon
T-Systems International GmbH
Bonn

Schlüsselworte:

Oracle, DB2, Oracle-Kompatibilitätsmodus, PL/SQL, SQL, ANSI-SQL, ANSI-SQL/PSM

Einleitung

Wer hat sich nicht schon über die Oracle-Lizenzpolitik geärgert (z.B. bei der Server-Virtualisierung)? Spätestens wenn neue Lizenzen benötigt werden oder ein neuer Supportvertrag verhandelt werden muss, stellt sich die Frage „Gibt es ernstzunehmende Alternativen zu Oracle?“.

Für relationale Datenbanken existiert seit vielen Jahren der ANSI-Standard ANSI SQL und für datenbankinterne Programmiersprachen entsprechend ANSI SQL/PSM. Wie es immer mit Standards ist, möchte sich natürlich jeder Hersteller von der Konkurrenz abheben und führt mehr oder weniger viele Erweiterungen bzw. Abweichungen zum Standard ein. Der Vorteil für die Hersteller liegt auf der Hand, je mehr vom Standard abweichende Funktionen genutzt werden, desto höher ist die Chance auf eine dauerhafte Kundenbindung. Die Aufwände für eine herstellerübergreifende Produktportierung steigen aber mit dem Einsatz von produktspezifischen Funktionen.

Eines der größten Hindernisse bei einem Wechsel von Oracle zu einem alternativen RDBMS ist damit die speziell für ein Oracle-RDBMS entwickelte Software.

Mit der Einführung von DB2 Version 9.7 hat IBM nun einen Oracle-Kompatibilitätsmodus bereitgestellt. Hiermit verspricht IBM, dass bestehende Programme für Oracle-Datenbanken ohne Änderung an DDL, DML und PL/SQL-Code unter DB2 lauffähig sind. In Marketingunterlagen wird gerne von 90-99% Kompatibilität gesprochen.

- Was ist nun von solchen Aussagen zu halten?
- Welche Funktionalitäten liegen in den verbleibenden 1-10%?

In den folgenden Kapiteln wird dieser Aussage an Beispielen aus der Praxis nachgegangen.
Es wird dabei aber nicht im Detail auf jedes Kompatibilitäts-Feature eingegangen. Es wird dabei von der Aussage der weitgehenden Oracle-Kompatibilität ausgegangen.

Kompatibilität ist aber nur ein Aspekt. Datenbank-Applikationen wollen auch entwickelt und betrieben werden. Auf Aspekte wie Know How in Entwicklung, Tuning und Betrieb etc. wird hier aber nicht weiter eingegangen.

Standardisierung

Wenn sich alle RDBMS-Hersteller an die ANSI-Standards ANSI SQL und ANSI-SQL/PSM halten würden, wäre ein Herstellerwechsel deutlich einfacher.

In den Standards kann man die einzelnen Aspekte grob in die Kategorien
- Datentypen (z.B. CHAR, INTEGER, TIMESTAMP, FLOAT)
- SQL-DDL-Syntax (z.B. CREATE TABLE tab (….))
- SQL-DML-Syntax (z.B. SELECT … FROM tab1 JOIN tab2 using (id))
- Standard-Funktionen (z.B. to_char)
- Interne Programmiersprache (z.B. PL/SQL, Java)
- Internes Verhalten (z.B. Datentypkonvertierung, mathematische Rundung)
- Transaktionsverhalten (z.B. Consistency modes, Lock-Verhalten)

unterteilen.

Applikationsseitig gibt es dann noch die Datenbankschnittstellen. Diese Schnittstellen stellen meist ein Abstraktionslayer zur Verfügung und sind somit meist schon datenbank-unabhängig (.z.B. JDBC, Perl-DBI).

Bedingt durch die DB-internen Programmiersprachen liegen aber auch Teile der Applikationen in der Datenbank, da spielen dann auch noch die bereitgestellten Standard-Funktionen, -Prozeduren, -Packages eine große Rolle.

Eine Menge Logik ist aber typischerweise in datenbank-externen SQL-Skripten hinterlegt, die im Falle Oracle z.B. mit SQLPlus ausgeführt werden. Im Gegensatz zu SQL ist SQLPlus aber nicht standardisiert.

Was ist eigentlich Oracle-Kompatibilität?

Bisher hat DB2 eigentlich nur den ANSI-SQL-Standard (und ANSI-SQL/PSM) implementiert.
Zum Einen unterstützt Oracle mittlerweile den ANSI-SQL Standard recht gut, diese Anteile sind zwischen DB2 und Oracle kompatible. Zum Anderen hat aber Oracle daneben immer noch die „alten“ nicht standardkonformen Funktionen etc. So kann man einen Outer-Join in Oracle entweder als

```
SELECT … FROM tab1 a, tab2 b WHERE a.id=b.id(+)
```

oder als

```
SELECT … FROM tab1 a RIGHT OUTER JOIN tab2 b ON (a.id=b.id)
```

schreiben.

Den deutlichsten Unterschied gibt es bei den Datentypen, In ANSI-SQL gibt es
- DATE (nur Datum)
- TIME (nur Uhrzeit)
- DATETIME (Datum + Uhreit)
- TIMESTAMP (Datum + Uhrzeit + Subsekunden + Timezone)

Oracle hat dagegen nur
- DATE (Datum + Uhrzeit)
- TIMESTAMP (Datum + Uhrzeit + Subsekunden + Timezone)

Das letzte Beispiel zeigt, dass es nicht damit getan ist den Syntax von Oracle zu unterstützten, sondern das man zwischen dem Oracle- und DB2-Verhalten umschalten können muss.

Frage am Rande: Wer hat sich nicht schon mal in Oracle den Datentyp TIME oder das Datum ohne Zeit gewünscht?

Ebenso ist das Transaktionsverhalten ist zwischen Oracle und DB2 sehr unterschiedlich, während in Oracle lesende Transaktionen nie schreibende Transaktionen blockieren, kann dies in DB2 auftreten (Repeatable Read).

Es muss in der Kompatibilität unterschieden werden in

- neue Funktionalität ohne Überschneidung mit bestehender Funktionalität (kommt neu dazu)
- bestehende Funktionalität muss geändert werden (dafür braucht man den Kompatibilitätsmodus)
- entfallende Funktionalität (da muss man nichts machen)

Bisher war die Kompatibilität auf die Schnittmenge von DB2 und Oracle beschränkt.

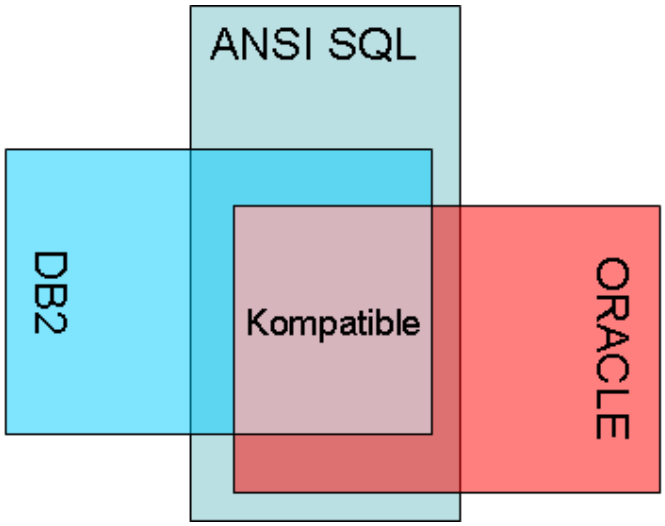

Abb. 1: Oracle-Kompatibilität DB2

Mit dem DB2 Oracle-Kompatibilitätsmodus erhöht sich der Anteil der Oracle-Konstrukte, die DB2 versteht deutlich

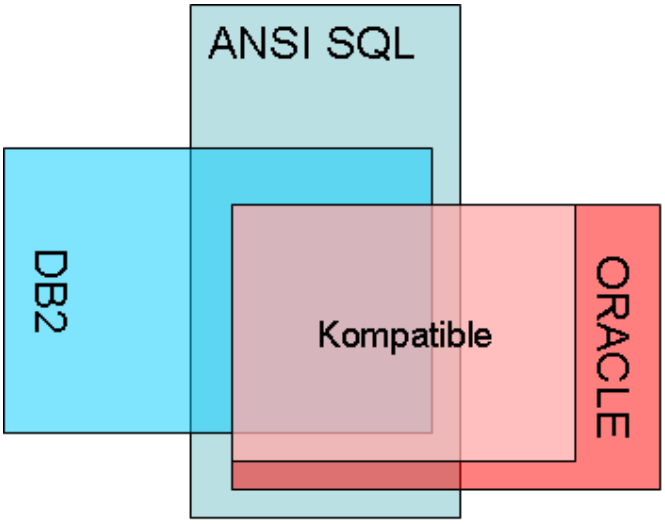

Abb. 2:Oracle- Kompatibilität DB2-Oracle-Kompatibilitätsmodus

Mit dem Oracle-Kompatibilitätsmodus versteht DB2 erstmals auch die Oracle-Nonstandard Funktionalitäten. Dies bezieht sich auf die Aspekte

- Bezeichner
- Datentypen
- DDL-Syntax
- DML-Syntax
- Transaktionsverhalten
- PL/SQL
- Systemviews
- Eingebaute PL/SQL-Packages

Im Folgenden wird auf einzelne Aspekte etwas genauer eingegangen.

Bezeichner

DB2 hatte normalerweise einige engere Beschränkungen in der Länge von Bezeichnern, dies wird mit dem Oracle-Kompatibilitätsmodus aufgehoben.

Kleine Fallen im Kompatibilitätsmodus:
Unter DB2 muss ein Indexname nur zur dazugehörenden Tabelle eindeutig sein, bei Oracle muss der Indexname im Schema eindeutig sein, Das kann Probleme beim Rückweg von DB2 nach Oracle machen. Ferner können in DB2 Bezeichener im Oracle-Kompatibilitätsmodus Längen bis 128 Zeichen haben, dahingegen kann Oracle hier nur bis 30 Zeichen.

Bei Bezeichnern, die andere Zeichen als A-Z0-9_ enthalten, muss man beim Tool clpplus (der DB2 SQLPlus-Ersatz) aufpassen. Während bei Oracle

```
CREATE TABLE XXX$TA_XXX;
DROP TABLE XXX$TA_XXX;
```

einfach so angegeben werden kann, referenziert bei clpplus das $ die Environment-Variablen. (hier die Variable TA_XXX). Ist ein $ Bestandteil des Bezeichners, muss dies über einen Escape-Character gequoted werden. Bezeichner in ,“’ funktionieren hier nicht wie erwartet.

Datentypen

Mit der Oracle-Kompatibilität unterstützt DB2 zusätzlich noch die Oracle-Datentypen wie

- NUMBER
- VARCHAR2, NCHAR, NVARCHAR, NCLOB,
- DATE (in der Art von Oracle)
- BOOLEAN
- TIMESTAMP(n)
- VARRAY, REFCURSOR

Ein kritischer Punkt ist die implizite Konvertierung von Datentypen (wie z.B. bei varcharcolumn=numbercolumn). DB2 ist normalerweise bei den arithmetischen und logischen Funktionen bei den Datentypen sehr streng. Ohne einen passenden CAST kann man da nicht einfach einem VARCHAR-Feld einen Zahlenwert zuweisen bzw. Einen VARCHAR mit einer Zahl vergleichen.

Im Oracle-Kompatibilitätsmodus verhält sich in diesem Punkt da DB2 genau so lasch wie Oracle (z.B. DATE+NUMBER oder NUMBER=VARCHAR2 liefern keinen Fehler).

Achtung:
Beim Datentyp NUMBER gibt es eine kleine Abweichung. Oracle unterstützt hier bis NUMBER(38), DB2 aber nur bis NUMBER(31). Diese Einschränkung wird funktional nicht auffallen, oft sind aber NUMBER-Spalten mit NUMBER(38) definiert worden, was in diesem Fall Änderungsaufwand bedeutet.

Es gibt aber kleine aber gemeine Abweichungen. Zum Beispiel werden in Oracle die impliziten Konvertierungen von DATE-Spalten mittels der Einstellung aus der Session für NLS_DATA_FORMAT vorgenommen. Diese Einstellungen auf Sessionebene existieren in dieser Form unter DB2 nicht.

Anmerkung:
Die implizite Typkonvertierung ist bei Oracle ja immer wieder für Probleme gut (z.B. Index wird nicht genutzt). Wer ist schon mal darüber gefallen, dass nach dem Ändern von NLS_LANG die Applikation auf einmal Fehler geschmissen hat? Selbst Oracle hatte da bis einschließlich 10g in DBMS_STATS einige Stellen, die nicht mit jedem NLS_LANG funktioniert hatten.

DDLs

DDLs liegen nicht im Fokus des DB2 Oracle-Kompatibilitätsmodus. DDL-Statements enthalten neben dem CREATE TABLE … oft noch Angaben für die Speicherung und Partitionierung.

Das reine CREATE TABLE, CREATE INDEX ist mit den Oracle-kompatiblen Datentypen unter DB2 nun 100% lauffähig

Partitionierung ist bei DB2 nicht 100% kompatible. Oracle bietet hier Partitionierung über RANGE, HASH, LIST und Subpartitions. DB2 unterstützt hier nur RANGE.

Ebenso gibt es Unterschiede bei den Storageparametern. Dies relativiert sich aber, da immer mehr dazu übergegangen wird, bei Oracle ASSM zu nutzen und somit sehr oft nur noch die reinen CREATE TABLE ohne Storageparameter enthalten sind. Dummerweise ist der Syntax für die Angabe von Tablespaces leicht unterschiedlich. Dieser lautet unter Oracle

```
CREATE TABLE  … TABLESPACE xxx
```

dagegen in DB2 aber

```
CREATE TABLE … IN xxx
```

Bei der Zuweisung von Tabellen zu Tablespaces muss man beachten, dass in DB2 eine Row immer in einen Block passen muss. Das bedeutet, dass bei breiten Tabellen ggf. ein Tablespace mit größerer Blocksize genommen werden muss. Hier zählt die maximale Breite.

Wenn Constraints etc. über ALTER TABLE angelegt werden lauert eine Inkompatibilität. Während Oracle

```
ALTER TABLE ADD (CONSTRAINT …
```

Erwartet (manchmal geht es auch ohne Klammern), müssen in DB2 die Klammern immer weggelassen werden

```
ALTER TABLE ADD CONSTRAINT …
```

Einen weiterer kleiner Unterschied gibt es, wenn zuerst Primary-Key-Indexe angelegt werden und dann mit ALTER TABLE der Primary Key-Constraint angelegt wird. Bei Oracle muss dann

```
ALTER TABLE tab ADD CONSTRAINT PK1 PRIMARY KEY(id)
USING INDEX idx1
```

genutzt werden, dagegen muss man bei DB2 nichts angeben. Falls es schon einen Index zu dem Constraint gibt, erhält man eine Warnung.

Mit dem DB2 Migration Toolkit werden diese Unterschiede automatisch umgesetzt. So ist nur noch in ganz wenigen Fällen Handarbeit angesagt.

Achtung eine Falle:
Bei den mehrspaltigen Primary Keys. erwartet Oracle, dass mindestens eine Spalte NOT NULL ist. DB2 erwartet hier auch im Oracle-Kompatibilitätsmodus dass alle Spalten NOT NULL sind. Dies sollte aber kein Problem sein, denn Primary Keys mit nullable Spalten machen normalerweise mehr Probleme als sie lösen (z.B. Joins sind da auf einmal nicht mehr ganz einfach).

Hinweis:
Bei DB2 ist der Begriff Partitionierung doppelt belegt. Zum Einen gibt es wie bei Oracle die Tabellenpartitionierung, zum Anderen gibt es bei DB2 noch das „Data Partitioning Feature (DPF), welches zur Verteilung von Daten über mehrere Server genutzt wird (Skalierung über Shared Nothing-Architektur). Die Syntax für die Partitionierung ist hier auch nicht kompatible. Oracle hat hier

```
PARTITION xxx VALUES LESS THAN …
```

dagegen nutzt DB2 das Konstrukt

```
PARTITION xxx STARTING … ENDING … EXCLUSIVE
```

Oracle-SQL-Syntax

Die SQL-Statements sind typischerweise in den Applikationen enthalten (wenn man mal PL/SQL-Packages außen vor lässt), sie stellen also die Schnittstelle zur Datenbank dar. Die Kompatibilität der SQL-Statments und der PL/SQL-Aufrufe ist somit auch die Mindestanforderung an einen Oracle-Kombatibilitätsmodus.

Oracle stellt einige Non-ANSI-SQL Syntaxkonstrukte bereit. Dies sind z.B.

- Table DUAL
- Pseudo-Spalten ROWNUM, ROWID
- TRUNCATE TABLE
- CREATE OR REPLACE
- MINUS SELECT (heisst in ANSI-SQL EXCEPT)
- (+) für outer join
- Parameterübergabe über Namen (para1=>123, para2=>'abc')

Die zusätzliche Unterstützung dieser Oracle-SQL-Konstrukte erhöht den Anteil der unter DB2 lauffähigen SQL-Statements sehr stark.

Anmerkung:
Neben TRUNCATE TABLE gibt es bei Oracle noch ALTER TABLE TRUNCATE PARTITION, dies wird unter DB2 nicht bereitgestellt.

Vordefinierte Funktionen

Neben der Unterstützung der Oracle-spezifischen-Syntax sind auch die bereitgestellten SQL-Funktionen für eine vollständige Kompatibilität wichtig. Im Oracle-Kompatibilitätsmodus werden

- Konvertierung TO_DATE, TO_CHAR, TO_NUMBER, …
- Berechnung ADD_MONTH, NEXT_DAY, Datumsarithmetik mit +
- String LPAD, RPAD, SUBSTR,…

bereitgestellt.
Die gängigen Oracle-Funktionen sind dabei enthalten. In den Tests wurden hierbei keine Funktionen vermisst.
Fehlende Funktionen lassen sich aber recht einfach als FUNCTION in DB2 implementieren, somit müssen die SQL-Statements, die diese Funktionen nutzen, nicht geändert werden müssen.

PL/SQL

Bei der PL/SQL-Kompatibilität gibt es die Punkte

- Kompatible Aufruf-Schnittstellen (Funktionsaufrufe, Parameter, Returnwerte,…)
- Syntax (Schleifen, IF, …)
- Funktionen (Eingebaute PL/SQL-Funktionen)
- Standard-Packages

Damit eine Applikation bei einer Umstellung nicht geändert werden muss, müssen die Schnittstellen identisch sein. Dies ist auch die Mindestanforderung an eine PL/SQL-Kompatibilität. Dies wird im Oracle-Kompatibilitätsmodus sehr gut erreicht.

Bei den Tests hat hier einzig die fehlende Unterstützung vom SUBTYPE Änderungsaufwand verursacht. Auch in PL/SQL unterstützt der Datentyp NUMBER nur 31 gültige Stellen. Die ergab an einer Stelle eine Änderung, da hier ein 37-stelliges Primzahlprodukt als Konstante benutzt wurde (was schon bei Oracle zu Problemen geführt hat).

Die Standard-Konstrukte wie Schleifen, Bedingungen etc. sind, soweit sie in den bestehenden PL/SQL-Code vor kamen, kompatibel.

Die größten Abweichungen gibt es im EXCEPTION-Handling. Dies ist hauptsächlich durch die unterschiedlichen Fehlercodes bedingt. Ferner ist das Konstrukt PRAGMA nicht kompatibel.

Zu den mitgelieferten Standard-Packages finden sich im nächsten Abschnitt weitere Informationen.

Datadictionary-Views/Systempackages

Die wichtigsten Datadictionary-Views wie DBA/ALL/USER_TABLES, DBA/ALL/USER_INDEXES, DBA_ROLE_PRIVS, etc. werden auch im Oracle-Kompatibilitätsmodus bereitgestellt.
Die Datadictionary-Views haben teilweise kleine Abweichungen in den Spalten.

Beispiel System-View ALL_OBJECTS:

Oracle 11.2.0.2	**DB2 9.7.4**
OWNER	OWNER
	OBJECT_SCHEMA
OBJECT_NAME	OBJECT_NAME
SUBOBJECT_NAME	
OBJECT_ID	OBJECT_ID
DATA_OBJECT_ID	DATA_OBJECT_ID
OBJECT_TYPE	OBJECT_TYPE
CREATED	CREATED
LAST_DDL_TIME	LAST_DDL_TIME
TIMESTAMP	TIMESTAMP
STATUS	STATUS
TEMPORARY	TEMPORARY
GENERATED	GENERATED
SECONDARY	
NAMESPACE	
EDITION_NAME	

Tab. 1: Kompatibilität der System-View ALL_OBJECTS

Es werden auch die etliche DBMS-Packages wie DBMS_JOB, DBMS_UTILITY, DBMS_SQL etc. bereitgestellt.

Wer intensiv Gebrauch von den System-Packages macht, wird hier ggf. einige Lücken finden (z.B. fehlt das doch recht nützliche DBMS_APPLICATION_INFO).

Transaktionsverhalten

Das Transaktionsverhalten von DB2 unterscheidet sich per Default deutlich von dem von Oracle. DB2 implementiert hier die Isolation-Levels aus ANSI-SQL, dahingegen implementiert Oracle eine Multi-Version-Read-Consistency.
Bei Oracle blockieren Sessions, die Datensätze lesen, nie die Sessions, welche die selben Datensätze ändern (Ausnahme SELECT FOR UPDATE). Hierfür erzeugt Oracle ggf. mehrere Versionen der gleichen Zeile. Der Standard-DB2-Modus stellt nur eine Version einer Zeile bereit. Bei Anwendungen, die nur kleine-schnelle Transaktionen haben, wird man hier bei DB2 keinen großen Unterschied zu Oracle sehen, dahingegen können parallel laufende große-langsame Transaktionen zu deutlichen Blockaden führen. Dafür werden große Transaktionen bei Oracle dann immer langsamer und enden ggf. im „ORA-01555 Snapshot too old“.

Der Oracle-Kompatibilitätsmodus ändert hier das Verhalten von DB2 auf das Oracle-Verhalten. Das funktioniert auch so wie erwartet.

Ferner gibt es einen großen Unterschied beim Locking von Datensätzen. Während Oracle immer nur einzelne Rows locked, hat DB2 einen Lock-Escalation-Mechanismus, d.h. wenn von einer Session zu viele Locks auf einer Tabelle gehalten werden, werden diese zu einem Table-Lock zusammen gefasst. Die Anzahl der gleichzeitigen Locks auf eine Tabelle kann dabei konfiguriert werden. Hier ist es ggf. notwendig unter DB2 die Zahl der Locks zu tunen.
Hier ändert der Oracle-Kompatibilitätsmodus nichts am Verhalten, ggf. muss hier also entsprechendes Performanz-Tuning gemacht werden, da die Default-Setzung des Parameters ggf. deutlich zu klein ist.

Synonyme

Es gibt im Oracle-Kompatibilitätsmodus auch Public Synonyme. Hier gibt es aber eine kleine Falle. Public Synonyms wie auch normale Synonyme funktionieren unter DB2 nur für Tabellen, Views und Sequences. Procedures, Functions und Packages werden nur über den Suchpfad gefunden. Der Default Suchpfad ist dabei

```
SET PATH=SYSTEM PATH, user
```

Es wird also zuerst in den System-Schemata gesucht und anschließend im Schema des aktuellen Users. Hier muss also ggf. direkt nach dem Login ein Statement zur Setzung des Pfades mit

```
SET PATH=SYSTEM PATH, schema1, schema2
```

eingefügt werden.

Beispiel Public Synonyme und PATH:

```
select * from tab1;
FEHLER bei Zeile 19:
SQL0204N  "UWE.TAB1" is an undefined name.

select func1(1) func1 from dual;
FEHLER bei Zeile 20:
SQL0440N  No authorized routine named "FUNC1" of type "FUNCTION" having
compatible arguments was found.

create public synonym tab1 for xx.tab1;
DB250000I: Der Befehl wurde erfolgreich ausgeführt.

create public synonym func1 for xx.func1;
DB250000I: Der Befehl wurde erfolgreich ausgeführt.

select * from tab1;
XX
----------
xx

select func1(1) func1 from dual;
FEHLER bei Zeile 28:
SQL0440N  No authorized routine named "FUNC1" of type "FUNCTION" having
compatible arguments was found.

set path=xx,current path;
DB250000I: Der Befehl wurde erfolgreich ausgeführt.

select func1(1) func1 from dual;
FUNC1
--------------------------------------------------
xx
```

Objektspeicherung

Die Speicherung der Objekte ist zwischen Oracle und DB2 nicht kompatible. Der größte Unterschied ist hierbei, dass bei DB2 – auch im Oracle-Kompatibilitätsmodus – eine Zeile immer in einen Block passen muss (LOBs sind hiervon ausgenommen). Hierbei zählt die maximale Zeilenlänge. Da benötigt man für Tabellen mit mehreren Freitext, Bemerkungs- bzw. Info-Feldern mit VARCHAR2(2000) schnell zusätzliche Tablespaces mit 8KB oder 16KB Blocksize.

Die Storageoptionen von Objekten sind bei DB2 zwar ähnlich zu Oracle, hier ist aber auf jeden Fall Handarbeit angesagt.

Tip:
Der einfachste Weg einmal einen Migrationstest von Oracle nach DB2 zu machen ist ein Strukturexport der Oracle-Schemata. Hier sollten beim Export die Storage- und Tablespace-Parameter nicht mitgeneriert werden.

SQLPlus

Zur Installation und ggf. auf auch für den Betrieb gibt es meist eine größere Menge von SQLPlus-Skripten. Diese enthalten neben den SQL-Statements meist auch noch SQLPlus-Anweisungen wie SPOOL, COLUMN etc.

Unter DB2 gibt es das **sqlplus**-kompatible Programm **clpplus.** Dies funktioniert für die gängigen Installations-/Patch-Skripte problemlos. Aus verständlichen Gründen konnte das Programm nicht sqlplus genannt werden. Wer die Oracle-Kompatibilität unter DB2 nutzen möchte wird wohl auch häufig einen Oracle-Client installiert haben und ggf. parallel nutzen wollen.

Das Gegenstück zur Oracle-Datei tnsnames.ora ist bei DB2 die XML-Datei db2dsdrivers.cfg. Dummerweise erlaubt der Connect mit clpplus bei Angabe der Datenbanknamen aus dieser Datei keinen Usernamen. Der Login erfolgt in diesem Fall immer mit dem aktuellen Betriebssystemuser. Diese Option ist also nur bedingt nützlich.

Die Gegenstücke zu den gängigen SQLPlus-Aufrufen

```
sqlplus username@dbname @script
sqlplus /nolog @script
```

(Es wird ja wohl keiner das Password in der Commandline übergeben) sind dann

```
clpplus –nw username@host:port/datenbank @script
clpplus –nw @script
```

In den Installationsskripten muss also der Aufruf von sqlplus durch den Aufruf von clpplus mit angepassten Parametern ersetzt werden.

In den Installationsskripten muss ferner ggf. auch die vorhandene Fehlerbehandlung überarbeitet werden. Statt ein `grep ORA- logfile.lst` muss dann ein

```
grep `^SQL` logfile.lst
```

gemacht werden.

Hinweis:
Hinter dem Kommando clpplus steckt ein Java-Programm. Das Starten dauert also einen Moment. Wenn auf einen Linux-Rechner X11 installiert ist öffnet clpplus normalerweise automatisch ein neues Fenster. Durch die Option –nw erfolgen Ein-/Ausgaben wie bei sqlplus über das aktuelle Fenster.

Auf die freie DB2-Version DB2-Express-C kann man mit clpplus leider nicht zugreifen. Der Versuch eines Zugriffes liefert dann den Fehler

```
DB250202E: Connections to databases of type DB2/LINUX
SQL09074 are not supported by this feature
```

Applikationsschnittstellen

Applikationen sprechen Oracle-Datenbanken meist über JDBC, ODBC, Embededd-SQL bzw. OCI an. Durch den OracleKompatibilitätsmodus funktionieren die JDBC und ODBC- basierten Applikationen ohne Code-Änderung. Hier muss nur die JDBC-URL bzw. die ODBC-DSN angepasst werden.

Es wird auch Embedded-SQL unterstützt. Der größte Unterschied fällt hier direkt beim Connect auf. Hier muss man unter Oracle

```
EXEC SQL CONNECT :user_name IDENTIFIED BY :password;
```

schreiben, wobei username den DB-Namen enthalten kann. Unter DB2 lautet der Connect aber immer

```
EXEC SQL CONNECT TO :dbname USERID :userid PASSWORD :password;
```

Hier muss immer Datenbank und Username getrennt sein. In den Generierungsskripten müssen aber auf jeden Fall die Oracle-Precompiler (proc) Aufrufe und die Oracle-Libraries, Include-, Link-Optionen gegen die DB2-Precompiler-Aufrufe (db2 precompile …) und die passenden DB2-Libraries, Include-, Link-Optionen ersetzt werden. Hier hatte sich der Test auf ein kleines Beispiel beschränkt, was aber nicht repräsentativ sein dürfte.

DB2 stellt auch ein OCI-kompatibles Datenbank-Interface bereit. Hier sind alle gängigen Funktionen der OCI-API implementiert. Hier müssen dann die Oracle-Header-Files durch die DB2-Headerfiles ersetzt werden (am Besten mit bedingter Übersetzung) und die Applikation muss gegen die DB2-Libraries gelinked werden. Applikationslogik ist hiervon nicht betroffen.

Die Chance, dass die Applikation beim Wechsel von Oracle auf DB2 mit Oracle-Kompatibilitätsmdus geändert werden muss, ist sehr gering. Bei Embedded-SQL und OCI hängt diese aber von den genutzten Funktionen der Schnittstellen ab. Bei den meisten Applikationen wird es hier mit Anpassen der Pfade, Headerfilename und einem Relink mit passenden Optionen getan sein.

Fehlerbehandlung

Jede Applikation sollte die ggf. auftretenden Datenbankfehler ordentlich behandeln (z.B. sollten bei Webapplikationen ja keine ORA- Meldungen direkt im Web-Frontend erscheinen).
Hierfür werden typischerweise die Oracle-Fehlercodes ausgewertet und gemäß der Applikationslogik behandelt. Hier wird normalerweise zwischen

- erwarteten Fehlern
  z.B.UNIQUE KEY-Violation bei Spalten deren Werte manuell eingegeben werden
- unerwarteten Fehlern
  z.B. ORA-00600, ORA-04030

unterschieden.

Die Fehlercodes sind dummerweise nicht zwischen Oracle und DB2 kompatible.

Beispiel unterschiedliche Fehler-Codes:

Fehler	**Oracle 11.2.0.2**	**DB2 9.7.4**
Tabelle nicht zugreifbar	-942 `ORA-00942: Tabelle oder View nicht vorhanden`	-204 `SQL0204N "UWE.YY" is an undefined name.`
Unique Violation	-1 `ORA-00001: Unique Constraint (UWE.XX_PK) verletzt`	-804 `SQL0803N One or more values in the INSERT statement, UPDATE statement, or foreign key update caused by a DELETE statement are not valid because the primary key, unique constraint or unique index identified by "1" constrains table "UWE.XX" from having duplicate values for the index key.`

Tab. 2: Fehler-Codes von SQL-Statements

Hier ist also auf jeden Fall Handarbeit angesagt (oder macht da etwa jemand keine Fehlerbehandlung?).

Security

Ein wichtiger Punkt in jeder Applikation sind die Benutzer und deren Berechtigungen. Hier unterscheidet sich DB2 deutlich von Oracle. Während Oracle per Default die Benutzer/Kennworte in der DB verwaltet, stützt sich DB2 auf bestehende Betriebssystem-Accounts. Auch das Kennwort wird dann gegenüber dem Betriebssystem geprüft.

DB2 unterscheidet im Gegensatz zu Oracle zwischen SCHEMA und Account. Ein Account ohne eigene Objekte braucht unter DB2 kein zugehörendes Schema. Ebenso benötigt ein Schema unter DB2 nicht zwangsweise einen zugehörenden Account, der Zugriff wird hier über Rechte am Schema erteilt.

Grants von Rechten auf Tabellen und Rollen funktionieren genauso wie bei Oracle. Es gibt aber Unterschiede bei den vordefinierten Rollen und System-Privilegien.

Integration von DB2 in bestehende Oracle-Umgebung

Typischerweise hat man ja nicht nur eine Datenbank. Wie sieht es nun aus, wenn man sowohl Oracle als auch DB2 nutzt?

Hier besteht normalerweise die Anforderung Daten aus einem System in ein anderes System zu übertragen. Hier müssen 3 Fälle unterschieden werden

- Highlevel (SOA etc.)
- Applikations Level (Applikation ließt Daten aus einer DB und schreibt in eine andere DB)
- Database Level (Datenaustausch direkt über Database-Links)

Die High-Level-Anbindung ist der einfachste Fall. Hier muss sich nur der Adapter an DB2 anmelden. Es gilt hier dann das im Kapitel „Applikationsschnittstellen“ gesagte.

Geschieht die Anbindung über eine Applikation muss diese sowohl zu Oracle als auch zu DB2 eine Verbindung offen halten können (meist gleichzeitig). Dies aus Sicht der DB-Schnittstellen möglich. Hier ist aber zu prüfen, ob das ggf. verwendete Database-Integration-Framework dies auch unterstützt (die mir bekannten können das).

Problematisch sind die Datenbanken, die über Database-Links kommunizieren. Wie die Kopplung auf Datenbankebene umgesetzt werden kann, hängt wesentlich von den technischen/fachlichen Anforderungen an die Kopplung ab. Hier hilft der Oracle-Kompatibilitätsmodus nur bedingt. Es gibt hier die Möglichkeiten

- Oracle Database Gateway for ODBC (dg4odbc)
- Oracle Heterogeneous Services
- DB2 Federation Server

Der dg4odbc ist eine einfache Möglichkeit von Oracle auf einen anderen Datenbanktypen zu zugreifen. Die Funktionalität ist aber beschränkt, z.B. werden keine verteilten Transaktionen unterstützt. Für Readonly Zugriffe wird dies aber ausreichen. Der dg4odbc muss nicht extra lizenziert werden. Mit dem Heterogeneous Services hat man alle Möglichkeiten der Verbindung von Oracle und DB2, diese Option ist aber auch lizenzpflichtig. Für den DB2 Federation Server gilt das Gleiche wie für den Oracle Heterogeneous Gateway, auch er muss extra lizenziert werden.

Die Applikationsanteile, die Database-Links nutzen, müssen also auf jeden Fall überarbeitet werden, da selbst der „einfache“ Database-Link-Syntax nicht mehr funktioniert.

Hinweis:
Mit dem dg4odbc funktioniert auch unter Unix/Linux, wenn auf dem Server zusätzlich unixodbc installiert ist. Dummerweise gibt es da den Oracle-Bug 7141047 „DG4ODBC CONNECTION FAILS WHEN USING UTF8“. Damit ist eine Kommunikation von einer Oracle-UTF8-Datenbank zu einer DB2-Datenbank leider nicht möglich.

Performanz

Wenn nun alle DDLs, DMLs und der PL/SQL-Code auch unter DB2 funktionieren, steht ja eigentlich einem Wechsel nichts mehr im Wege. Ja, wenn da nicht die Performanz wäre. In der bestehenden Applikation wurde ggf. einiges für die Performanzoptimierung implementiert. Hierunter fallen z.B.

- Spezielle Objektspeicherung (Partitionierung, Indexe etc.)
- spezifische DBMS_STATS- Aufrufe
- Stored Outlines / SQL Plan Management
- Optimizer-Hints

In DB2 gibt es jetzt aber keinen Oracle-Optimizer-Kompatibilitätsmodus. Im Idealfall muss man hier aber gar nichts machen. Hier ist also erstmal Testen, Testen, Testen angesagt, natürlich mit einem der Produktion ähnlichen Datenvolumen und Datenverteilung. Am Ende hat man dann eine Liste von SQL-Statements, die optimiert werden müssen. Das entspricht genau dem, was auch bei einem Wechsel des Oracle-Releases passiert. Die Werbung sagt ja immer das beim Release-Wechsel alles schneller wird, die Erfahrung zeigt aber, dass oft auch das Gegenteil eintritt.

Bei der Optimierung muss man sich also etwas umgewöhnen, die Werkzeuge sind aber sehr ähnlich. Leider kann man die Erfahrungen aus dem Optimieren von Oracle-Statements nur teilweise übernehmen.

DBMS_STATS

Das DB2 Gegenstück zu DBMS_STATS ist RUN_STATS. Dies Kommando erzeugt unter DB2 die Optimizer-Statistiken. Die Funktionalität ist hier ähnlich zu DBMS_STATS.

Stored Outlines/SQL Plan Management

Das Gegenstück zu den Stored-Outlines sind bei DB2 die Optimization-Profiles. Dies sind XML-Dokumente, die einem Statement-Text die zugehörenden „Optimizer-Hints“ zuordnen.
Beispiel:

```
<OPTPROFILE VERSION=“9.7.4“>
 <STMTPROFILE ID=“beispiel“>
  <STMTKEY>
   <![CDATA SELECT * FROM test_tab WHERE id=1]]>
  </STMTKEY>
  <OPTGUIDELINES>
    <IXSCAN TABLE=’test_tab’ INDEX=’ind’/>
  </OPTGUIDELINES>
 </STMTPROFILE>
>/OPTPROFILE>
```

Optimizer-Hints

Entgegen der weitläufigen Meinung beherrscht DB2 auch Optimizer-Hints.

DB2 bietet hier 2 Wege um dem Optimizer Hinweise zu geben.
Zum Einen als Bestandteil des Syntax der SQL-Statement (was in ANSI-SQL beschrieben ist)

```
SELECT .. OPTIMIZE FOR n ROWS
SELECT .. FETCH FIRST n ROWS ONLY
SELECT .. WHERE col=0 SELECTIVITY 0.0001
```

Hier erhält man entsprechend einen Syntaxfehler, wenn der Hint nicht korrekt ist.

Zum anderen für komplexe Hints als Kommentar im SQL-Statement. Deren Syntax ist ein XML-Ausdruck (die OPTGUIDELINES aus dem OPTPROFILE von oben) und somit unterschiedlich zur Syntax der Oracle-Optimizer-Hints.
Beispiel Oracle-Optimizer-Hint.

```
/*+ INDEX(tab ind) */
```

DB2-Optimizer-Hint

```
/* <OPTGUIDELINES>
  <IXSCAN TABLE='tab' INDEX='ind'/>
</OPTGUIDELINES> */
```

Hinweis:
Wer hat nicht schon mal einen Fehler in einem Oracle-Optimizer-Hint gehabt (Indexname falsch geschrieben, Tabellenname statt Alias, etc.) und dann ewig lange danach gesucht? Oracle akzeptiert ja nun mal alles im /*+ … */ und wenn es fehlerhaft ist, wird es einfach ignoriert.

Und was macht DB2?
Bei fehlerhaften Optimizer-Hints liefert DB2 eine Warnung (Wer hat sich das nicht schon bei Oracle gewünscht?).

```
SELECT *
FROM test_tab
WHERE id=123
/*<OPTGUIDELINES> <IXSCAN TABLE='tab' INDEX='ind'/> </OPTGUIDELINES>*/
;
ID      TEXT
------- ---------
SQL0437W  Performance of this complex query may be sub-optimal.  Reason
code: "13". .. SQLSTATE=01602

0 record(s) selected with 1 warning message printed
```

Betrieb

Wenn man nun eine Oracle-Datenbank im DB2 Oracle-Kompatibilitätsmodus lauffähig und getestet hat, kommt der als letzter Schritt noch der produktive Betrieb. Da stehen dann Fragen wie

- Backup/Restore
- Monitoring
- Datenbank-Reorganisation
- Datenbank-Tuning
- Fehleranalyse
- Betriebs-Know-How

- …

auf dem Programm. Bei denen hilft dann aber der Oracle-Kompatibilitätsmodus nicht weiter. Um ein Datenbanksystem ordentlich betreiben zu können (wie auch dafür entwickeln zu können) muss man es beherrschen. Hierzu muss man sich wohl oder übel mit den Details von DB2 beschäftigen. Einfacher ist es da ggf. für den, der seinen Datenbank-Betrieb extern machen lässt.

Beim Betrieb stellt sich dann ja auch die Frage „Können wir mit einer anderen Datenbank nicht auch die Kosten senken?“. Zum Kosten sparen muss einfach eine „kritische“ Masse an Systemen erreicht werden. Wenn man nur eine DB unter DB2 laufen lassen möchte und Dutzende weiterhin unter Oracle wird sich damit noch kein „Business Case“ ergeben.

Erfahrungen mit der DB2-Kompatibilität bei real existierenden Anwendungen

Im Folgenden werden die Erfahrungen bzgl. der DB2-Oracle-Kompatibilität am Beispiel bestehender Datenbanken dargestellt.

*Hinweis***:** Die Ergebnisse sind dabei nicht repräsentativ und werden hier nur exemplarisch gezeigt. Das Ergebnis kann bei anderen Anwendungen deutlich anders aussehen.

Plain-SQL Datenbank

Plain-SQL-Datenbanken (also Datenbanken ohne PL/SQL) sind der einfache Fall, da die gängigen SQL-Syntax-Varianten und die gängigen SQL-Funktionen auch in DB2 vorhanden sind.

Objekttyp	**Anzahl**	**Erfolgreich**	**Kompatibilität**	**Kommentar**
TABLE	17	16	94%	RAW Spalte
INDEX	26	26	100%	
VIEW				
SEQUENCE	17	17	100%	
TRIGGER				
PACKAGE				
FUNCTION				
PROCEDURE				
TYPE				
CONTRAINTS				
Others	0	0		
Application-SQLs	103	103	100%	
Total	**163**	**162**	**99%**	

Tab. 3: Kompatibilität einer Web-Applikation

Diese Applikation greift mit einem Applikationsaccount auf die DB zu und die Useraccounts wurden intern über Tabellen verwalten. Die SQLs wurden dabei aus der v$sqlarea gesammelt.

Die Constraints waren hier direkt in den CREATE TABLE Statements enthalten.

Diese Kategorie von Applikationen ist ideal für den Oracle-Kompatibilitätsmodus.
Die eine „nicht kompatible“ Tabelle, enthält dabei eine RAW Spalte. Die Spalten mit RAW(n) müssen dann auf BLOB(n) geändert werden

Datenbanken mit PL/SQL

Das andere Extrem bzgl. der Kompatibilität sind Datenbanken, deren Applikationsschnittstelle eine PL/SQL-Abstraktionsschicht ist.

Objekttyp	Anzahl	Erfolgreich	Kompatibilität	Kommentar
TABLE	168	164	98%	Syntax PARTITION
INDEX	221	215	97%	Syntax PARTITION
VIEW	1	1	100%	
SEQUENCE	80	80	100%	
TRIGGER	56	56	100%	
PACKAGE	133	6	**5%**	Mit SUBTYPE
		116	87%	Ohne SUBTYPE
FUNCTION	3	0	**0%**	DBMS_APPLICATION_INFO
		3	100%	Ohne
PROCEDURE	31	2	6%	DBMS_APPLICATION_INFO
		31	100%	Ohne
TYPE	7	7	100%	
CONTRAINTS	312		**0%**	ADD (…) Syntax
		312	100%	Ohne ( )
Others	0	0		
Application-SQLs	357	357	100%	Nur PL/SQL-Aufrufe
Total	**1369**	**888**	**65%**	**Ohne Anpassungen**
		1342	**98%**	**Leichte Anpassungen**

Tab.4: Kompatibilität einerPL/SQL-Applikation

Diese Applikation nutzt Oracle-DB-Accounts. Die User-Verwaltung ist in den PL/SQL-Packages enthalten. Diese Packages ließen sich zwar übersetzen, die darin als Dynamic-SQL enthaltenen CREATE/ALTER/DROP USER Statements sind aber nicht lauffähig.

Bei PACKAGE, PROCEDURE, FUNKTION, CONSTRAINTS ist oben aufgeführt, wie sich die Kompatibilität ändern würde, wenn die entsprechende Syntax unterstützt wird. Hier zeigt sich sehr deutlich, dass 99% Kompatibilität nach Features nicht auch heißt, dass 99% der Applikation durch die Kompatibilität abgedeckt sind.

Die Tabellenobjekte inkl. Index, Trigger, Constraints etc ließen sich aber zu 100% über das DB2 MigrationToolkit anlegen.

Die nicht übersetzbaren Packages enthalten die nicht unterstützten Konstrukte

- DBMS_APPLICATION_INFO.SET_MODULE
- DBMS_LOCK.SLEEP, DBMS_OBFUSCATION_TOOLKIT.DESENCRYPT
- DBMS_SHARED_POOL.KEEP, DBMS_STATS.IMPORT_TABLE_STATS
- DBA_JOBS, USER_JOBS, DBA_TAB_STATISTICS
- V$SESSION
- SELECT FOR UPDATE NOWAIT
- PRAGMA …

Hinweis:
Eine Testapplikation ließ sich erstmal nicht installieren, da das Namenskonzept Bezeichner der Form AAA$BB_CCC vorsah. Das Program clpplus behandelt aber $BB_CCC an einigen Stellen als Environment-Variable BB_CCC. Durch ein Global-Replace von „$“ nach „_“ lies sie sich dann ohne Probleme auch unter DB2 installieren.

Mit dem DB2-MigrationToolkit hatte die Installation hier dann ohne weitere Änderungen funktioniert.

Kompatibilität

Zusammenfassend stellt dich die Oracle-Kompatibilität von DB2 wie folgt dar.

Anteil	**Erfüllung**	**Kommentar**
SQL	++	
PL/SQL	+	
DDL-Statements	+/-	Abweichungen Storage clauses etc.
Fehlercode	-	
Transactionsverhalten	++	
Database Links	-	Muss überarbeitet werden
Applikationsschnittstelle	+	Ggf. Recompile/Relink der Applikation
SQLPlus	+/-	OK für DDL-Skripts, komplexe Skripts nicht
Performanz	+/-	Das Performanzverhalten vonDB2 ist anders als bei Oracle.
Performanztuning	-	Hier geht’s nicht ohne DB2 Know-How
Security	+/-	Anderes Schema/User Konzept
Integration	+/-	Ggf. deutliche Anpassungen notwendig
Betriebliche Aspekte	-	Hier geht’s nicht ohne DB2 Know-How

Tab.5: Aspekte der Kompatibilität

Fazit

Um eine bestehende Oracle-DB auf den Oracle-Kompatibilitätsmodus von DB2 umzustellen, ist es nicht nur mit der Applikationsumstellung getan. Um die reine Technik um herum gibt es noch etliche Punkte, bei denen Änderungen notwendig sind.

Der Oracle-Kompatibilitätsmodus ist eigentlich erstaunlich „kompatibel“, auch wenn das ein oder andere nicht 100% kompatibel ist. IBM hat hier aber auch keine 100% Kompatibilität versprochen. Die Tests haben gezeigt, dass das Versprechen >90% erreichbar ist. Je nach Applikation können aber die wenigen verbleibenden „nicht kompatiblen“ Prozente sehr viel Arbeit machen.

Der Oracle-Kompatibilitätsmodus ist besonders für Software-Hersteller ein interessantes Feature. Er vereinfacht die Nutzung von „Oracle-Datenbank-Anwendungen“ unter DB2, erlaubt die Weiternutzung von umfangreichen PL/SQL-Code und kann somit die „Hemmschwelle für den Einsatz von DB2 senken. Der Hauptaspekt der Oracle-Kompatibilität ist hierbei, dass die Schnittstellen zur Applikation (aufgerufene SQLs bzw. PL/SQLs) nicht geändert werden müssen.

An den ausgewählten Beispielen sieht man, dass „Kleinigkeiten“, die nicht durch den Oracle-Kompatibilitätsmodus abgedeckt sind, dazu führen können, dass für diese Applikation der Kompatibilitätsgrad gering werden kann.

Insbesonders wenn in PL/SQL „Basis-Packages“ von allen anderen Packages referiert werden, führt eine Inkompatibilität in den Basis-Packages schnell dazu, dass der PL/SQL-Anteil unter DB2 ohne Änderung überhaupt nicht funktioniert.

Wie gut dies bei einer vorhandenen Oracle-Anwendung funktioniert, kann aber nur durch einen individuellen Test festgestellt werden.

Der Oracle-Kompatibilitätsmodus ist gut geeignet für
- Plain-SQL-Applikationen
- Wenn die Performanz unter Oracle auch ohne große Optimierung bzgl Storage-Parametern/Partitionierung etc. erreicht wurde
- Wenn nicht alle PL/SQL-Features ausgenutzt werden

Trotz Oracle-Kompatiblilitätsmodus muss man
- genau prüfen, ob alle genutzten Oracle-Features unter DB2 auch vorhanden sind
- sehr gut Testen, da „unsaubere Programmierung“ unter Oracle sich unter DB2 ggf. etwas anders verhält.

Der Oracle-Kompatibilitätsmodus von DB2 vereinfacht
- die Migration von Oracle auf DB2 sehr deutlich
- für Softwarehersteller die Unterstützung von DB2 (zusätzlich zu Oracle)

Trotz Oracle-Kompatibilitätsmodus muss
- die Benutzer/Berechtigungsverwaltung der Applikationen ggf. angepasst werden
- weiterhin Performanztuning für DB2 beherrscht werden
- Betriebserfahrung für DB2 vorhanden sein (Monitoring, Backup/Restore etc.)

Mit clpplus lassen sich die meisten Oracle-Installationsskripte unverändert ausführen.
Da diese SQL-Skripte aber meist aus Shellskripten aufgerufen werden, müssen dann in diesen Shellskripten Anpassungen bei Aufruf und Fehlerauswertung vorgenommen werden.

Wie nicht anders zu erwarten war, ist aber immer noch etwas „Handarbeit“ angesagt.

Getting Started

Für eine Überprüfung der Kompatibilität gibt es von IBM ein Utility „MeetDB2“, welches über den Support erhältlich ist. Dieses Utility prüft eine bestehende Oracle-Datenbank (man füttert es mit den SQL-Skripten, die die DB anlegen) auf DB2-Kompatibilität. Die Meldungen sind dabei recht aussagekräftig.

Die meisten Oracle-Kompatibilitätsfeatures (bis auf PL/SQL) können schon mit der freien Version DB2-Express-C ausprobiert werden. Der Download ist unter http://www-01.ibm.com/software/data/db2/express/download.html möglich (nach Registrierung). Dort gibt es auch eine „Virtual-Appliance“ für VMWare. Daneben gibt es auch Cloud-Dienste mit DB2-Express-C .

Eine Zusammenfassung bzw. Übersicht über den Oracle-Kompatibilität-Modus kann man unter http://www.ibm.com/developerworks/data/library/techarticle/dm-0907oracleappsondb2/ finden.

Eine einführendes kostenloses Buch „Getting Started with DB2 Express-C“ kann man unter http://www.ibm.com/developerworks/wikis/display/DB2/FREE+Book-+Getting+Started+with+DB2+Express-C finden.

Hinweis:
Wer zum Testen der Oracle-Kompatibilität einen Schema-Struktur-Export über generierte SQL-Skripte nutzt, die alle Storage-Optionen enthalten, wird feststellen, dass die meisten Tabellen und Indexe sich nicht anlegen lassen. Ferner müssen unter DB2 Tablespaces mit den passenden Blockgrößen bereitgestellt sein, damit die breiten Tabellen (nach maximaler Anzahl Bytes) auch installierbar sind.

Next Steps

Die Hemmschwelle für seine Applikationen neben Oracle auch DB2 zu unterstützen, wird durch den Oracle-Kompatibilitätsmodus deutlich reduziert. Die wichtigsten Oracle-spezifischen Konstrukte werden nun auch von DB2 unterstützt. Da ferner DB2 wie auch Oracle den ANSI-SQL-Standard implementieren, hat sich die Schnittmenge der gemeinsam unterstützten Features deutlich vergrößert. Da DB2 keine komplette Implementierung von Oracle bereitstellt, wäre der optimale Weg zur parallelen Unterstützung von Oracle und DB2 bei der Applikationsentwicklung die Entwicklung unter DB2 mit eingeschalteter Oracle-Kompatibilität zu machen. Hierbei ist die Erwartungshaltung, dass dann unter Oracle keine (bzw. sehr wenige) nicht unterstützten Features auffallen.

Damit hätte man dann nur einmal die Entwicklung, aber weiterhin zweimal Testen und zweimal Performanztuning. Man sollte auch nicht verschweigen, dass dann der Support auch in der Lage sein muss, die Applikation unter beiden Datenbanken zu unterstützen.

Quellen

- Oracle to DB2 Conversion Guide: Compatibility Made Easy
  ibm.com/redbooks Nov 2000
- DB2 9.7: Run Oracle applications on DB2 9.7 for Linux, UNIX, and Windows
  ibm.com/developerworks May 2011
- Manual IBM DB2 Database for Linux, Unix and Windows ibm.com/db2/
  ibm.com/developerworks May 2011
- Getting Started with DB2 Express-C
  ibm.com/developerworks 2009
- Manual Oracle Database 11*g* Release 2 (11.2) Documentation
  http://www.oracle.com/technetwork/database/enterprise-edition/overview/index.html

Kontaktadresse:

Uwe Simon
T-Systems International GmbH
Am Propsthof 49
D-53121 Bonn

Telefon:	+49 (0) 228 181 42182
Fax:	+49 (0) 228 181 42172
E-Mail	uwe.simon@t-systems.com
Internet:	www.t-systems.de