Multi-temporale Datenbank Features in Oracle 12c

Philipp Salvisberg
Trivadis AG
Glattbrugg (Zürich)

Schlüsselworte
Oracle 12c, Temporal Validity, Temporal Flashback Query, Flashback Data Archive, Total Recall, Bitemporalität

Einleitung

Oracle 12c hat ein neues Feature namens "Temporal Validity". Mit Temporal Validity können eine oder mehrere Zeitdimensionen zu einer Tabelle hinzugefügt werden. Zur Bestimmung des Periodenbeginns und -endes lässt Oracle existierende oder auch automatisch erzeugte Columns zu. Oracle bietet in 12c - kombiniert mit Flashback Data Archive - native Bitemporalität und sogar Multi-Temporalität an. Dieser Vortrag erklärt anhand von Beispielen die unterschiedlichen Historisierungsarten, wie diese anzuwenden sind und positioniert dabei die neuesten 12c Features. - Eine perfekte Gelegenheit ein tri-temporales Datenmodell live in Aktion zu sehen.

Semantik und Granularität von Perioden

Oracle definiert in Flashback Data Archive Perioden mit einem halboffenen Intervall. D.h. ein Zeitpunkt x gehört zu einer Periode, wenn x >= dem Periodenbeginn und x < dem Periodenende ist. Es verwundert also nicht, dass Oracle auch für Temporal Validity halboffene Intervalle verwendet. Die nachfolgende Abbildung verdeutlicht das Prinzip an einem Beispiel:

Abb. 1: Beispiel zwei aufeinanderfolgender Perioden

Der Vorteil von halboffenen Intervallen liegt darin, dass das Ende des vorangegangenen Intervalls identisch mit dem Anfang des nachfolgenden Intervalls ist. Dadurch entsteht keine Lücke und die Granularität einer Periode (Jahr, Monat, Tag, Sekunde, Millisekunde, Nanosekunde) ist irrelevant. Der Nachteil ist, dass die Formulierung einer Einschränkung auf einen Zeitpunkt mit einer SQL Expression langatmiger als bei einem geschlossenen Intervall ist, da keine BETWEEN Condition verwendet werden kann.

Des Weiteren verwendet Oracle NULL für -∞ und +∞. Die Where Klausel zur Einschränkung der jetzt gültigen Intervalle sieht beispielsweise wie folgt aus:

```
WHERE (vt_start IS NULL OR vt_start <= SYSTIMESTAMP)
  AND (vt_end IS NULL OR vt_end > SYSTIMESTAMP)
```

Verwendung temporaler Perioden

In einem Entity-Relationship-Modell können temporale Perioden für Stammdaten und Referenzdaten verwendet werden. Transaktionen und Positionen benötigen hingegen keine temporalen Perioden, da diese Daten schon einen expliziten Zeitbezug - nämlich einen oder mehrere Zeitstempel - haben.

Korrekturen werden hier ähnlich wie in einer Buchhaltung mit Hilfe von Stornos oder Korrektur-/Differenzbuchungen durchgeführt.

In einem dimensionalen Modell ist es ähnlich. Dimensionen entsprechen hier Stamm- und Referenzdaten und haben bei einer SCD2 Modellierung entsprechend eine temporale Zeitdimension. Fakten werden wie Transaktionen und Positionen im Entity-Relationship-Modell nicht mit temporalen Perioden ergänzt. Stattdessen werden bei Fakten ein oder mehrere Beziehungen zur Zeitdimension abgebildet. Korrekturen erfolgen auch hier über Stornos und Korrektur-/Differenzbuchungen.

Transaktionszeit (Transaction Time - TT)
Die Transaktionszeit oder auch Systemzeit ist mit einem Flugschreiber vergleichbar. Der Flugschreiber zeichnet alle Veränderungen auf, so dass der aktuelle Zustand und die Vergangenheit zu jedem Zeitpunkt nachvollzogen werden kann. Änderungen in der Vergangenheit sind keine möglich. Änderungen in der Zukunft ebenfalls nicht.

Beispiel: Die Änderung der Job Bezeichnung von "Analyst" auf "Manager" wird im System am 15.04.2013 um 15:42:42 erfasst. Die vorherige Bezeichnung Analyst wird auf diesen Zeitpunkt terminiert und die neue Bezeichnung Manager ist ab diesem Zeitpunkt gültig.

Oracle unterstützt die Transaktionszeit mit Hilfe von Flashback Data Archive (auch bekannt als Total Recall). Flashback Data Archive erlaubt es, Daten aus der Vergangenheit konsistent darzustellen.

SCN	Session A	Session B
1	INSERT INTO emp (empno, ename, job, sal, deptno) VALUES (4242, 'CARTER', 'CLERK', '2400', 20);	
2	SELECT COUNT(*) FROM emp; -- 15 rows	
3		SELECT COUNT(*) FROM emp; -- 14 rows

Abb. 2: Konsistente Sicht der Vergangenheit

Wie viele Rows liefert die Abfrage "SELECT COUNT(*) FROM emp AS OF SCN 3;" basierend auf der Abbildung 2? - 14 Rows. Dies ist eine gute und vernünftige Implementation auf Datenbankebene, zeigt aber, dass eine konsistente Darstellung der Vergangenheit eine Definitionssache ist und in diesem Fall nicht die Sicht der Session A wiederspiegelt.

Gültigkeitszeit (Valid Time - VT)
Die Gültigkeitszeit beschreibt den Zeitraum wann etwas in der realen Welt als gültig betrachtet wird. Dieser Zeitraum ist unabhängig von der Eingabe im System und muss aus diesem Grund explizit gepflegt werden. Änderungen und Abfragen sind zu jedem Zeitpunkt möglich, d.h. in der Vergangenheit wie auch in der Zukunft.

Beispiel: Die Änderung der Job Bezeichnung von "Analyst" auf "Manager" wird am 01.01.2014 gültig. Die vorherige Bezeichnung Analyst wird auf diesen Zeitpunkt terminiert und die neue Bezeichnung Manager ist ab diesem Zeitpunkt gültig. Wann diese Eingabe im System erfolgt, ist in diesem Fall unwichtig.

Entscheidungszeit (Decision Time - DT)
Die Entscheidungszeit beschreibt den Zeitpunkt wann eine Entscheidung gefällt worden ist. Dieser Zeitpunkt ist unabhängig von der Erfassung in einem System und auch unabhängig davon, wann die Entscheidung in Kraft tritt. Änderungen in der Vergangenheit sind möglich. Änderungen in der Zukunft hingegen nicht.

Beispiel: Die Änderung der Job Bezeichnung von "Analyst" auf "Manager" wurde am 24.03.2013 beschlossen. Die vorherige Bezeichnung Analyst wird auf diesen Zeitpunkt auf der Entscheidungszeitachse terminiert und die neue Bezeichnung Manager ist ab diesem Zeitpunkt auf der Entscheidungszeitachse gültig. Wann diese Eingabe im System erfolgt und wann der Mitarbeiter sich Manager nennen darf, ist in diesem Fall nicht relevant.

Historisierungsarten
Die Unterscheidung der Historisierungsarten basiert einerseits auf den Zeitdimensionen gemäss Abbildung 3 sowie der Kombination aus den verschiedenen Zeitdimensionen.

Abb. 3: Zeitdimensionen

Nicht-temporale Modelle beinhalten keine Zeitdimensionen (z.B. EMP und DEPT in Schema SCOTT). Uni-temporale Modelle verwenden nur eine Zeitdimension (z.B. Transaktionszeit oder Gültigkeitszeit). Bitemporale Modelle verwenden genau zwei Zeitdimensionen (z.B. Transaktionszeit und Gültigkeitszeit). Multi-dimensionale Modelle verwenden mindestens drei Zeitdimensionen. Tri-dimensionale Modelle basierend auf genau drei Zeitdimensionen.

Temporal Validity
Das Feature Temporal Validity umfasst die DDL und DML Erweiterungen in Oracle 12c im Bereich temporaler Datenhaltung. CREATE TABLE, ALTER TABLE und DROP TABLE sind um eine PERIOD FOR Klausel erweitert worden. Hier ein Beispiel:

```
SQL> ALTER TABLE dept ADD (
  2     vt_start DATE,
  3     vt_end   DATE,
  4     PERIOD FOR vt (vt_start, vt_end)
  5  );

SQL> SELECT * FROM dept;

    DEPTNO DNAME          LOC           VT_START   VT_END
---------- -------------- ------------- ---------- ----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

VT bezeichnet die Periode und ist eine hidden Column. Die Zuordnung der VT Periode zur VT_START und VT_END ist im Oracle Data Dictionary in der Tabelle SYS_FBA_PERIOD festgehalten. Auch wenn nur genau eine PERIOD FOR Klausel angegeben werden kann, ist es möglich durch weitere ALTER TABLE Statements zusätzliche Perioden zu definieren.

Beim Anlegen einer Periode wird im Hintergrund auch ein einfacher Constraint definiert, der negative Zeiträume verhindert (VT_START > VT_END). Allerdings ist es nicht möglich einen Objekt-Identifier zu definieren und damit sind kein temporalen Constraints möglich (z.B. Verhindern von überlappenden Perioden, Lücken in Perioden, verwaisten Parent- oder Child-Perioden).

Im Bereich DML liefert Oracle 12c noch keine Unterstützung. Gewünscht wären hier beispielsweise temporale Insert, Updates und Deletes; Änderungen eines Subset von Columns über eine Periode; Merge von aufeinanderfolgenden Zeiträumen mit identischem Inhalt (z.B. nach Änderungen). D.h. eine temporale Änderung muss wie im nachfolgenden Beispiel manuell erfolgen:

```
SQL> UPDATE dept SET vt_end = DATE '2014-01-01' WHERE deptno = 30;
SQL> INSERT INTO dept (deptno, dname, loc, vt_start)
 2        VALUES (30, 'SALES', 'SAN FRANCISCO', DATE '2014-01-01');

SQL> SELECT * FROM dept WHERE deptno = 30 ORDER BY vt_start NULLS FIRST;

 DEPTNO    DNAME          LOC           VT_START   VT_END
---------- -------------- ------------- ---------- ----------
        30 SALES          CHICAGO                  2014-01-01
        30 SALES          SAN FRANCISCO 2014-01-01
```

Temporal Flashback Query
Das Feature Temporal Flashback Query umfasst die Erweiterung in Oracle 12c betreffend temporaler Abfragen von temporalen Daten. Oracle hat die bestehenden Flashback Query Interfaces erweitert. Die FLASHBACK_QUERY_CLAUSE im SELECT Statement wurde um eine PERIOD FOR Klausel ergänzt. Hier ein Beispiel:

```
SQL> SELECT *
  2    FROM dept AS OF PERIOD FOR vt DATE '2015-01-01'
  3   ORDER BY deptno;

    DEPTNO DNAME          LOC           VT_START   VT_END
---------- -------------- ------------- ---------- ----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          SAN FRANCISCO 2014-01-01
        40 OPERATIONS     BOSTON
```

Statt "AS OF" kann auch "VERSIONS" wie bei Flashback Query verwendet werden, allerdings ist zu beachten, dass nur eine Klausel gleichzeitig verwendet werden kann. D.h. falls mehr als eine temporale Periode verwendet wird, sind die Filterkriterien in der WHERE Klause zu definieren.

Analog zu Flashback Data Archive ist die PERIOD FOR Klausel für Views nicht anwendbar. In diesem Fall sind die Erweiterungen im Bereich des PL/SQL Packages DBMS_FLASHBACK_ARCHIVE interessant, welches um die Prozeduren ENABLE_AT_VALID_TIME und DISABLE_ASOF_VALID_TIME erweitert worden ist, um einen temporalen Kontext zu managen. Hier ein Beispiel:

```
SQL> BEGIN
  2     dbms_flashback_archive.enable_at_valid_time(
  3        level      => 'ASOF',
  4        query_time => DATE '2015-01-01'
  5     );
  6  END;
  7  /

SQL> SELECT * FROM dept ORDER BY deptno;

    DEPTNO DNAME          LOC           VT_START    VT_END
---------- -------------- ------------- ---------- ----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          SAN FRANCISCO 2014-01-01
        40 OPERATIONS     BOSTON
```

Diese Erweiterungen erlauben es aber nicht, die temporale Periode anzugeben, d.h. sofern mehr als eine temporale Periode für eine Tabelle definiert ist (z.B. Gültigkeitszeit und Entscheidungszeit), empfiehlt es sich die Kontexte ausschliesslich über die WHERE Klausel zu setzen.

Eine Limitation von Oracle 12c ist, dass die Temporal Flashback Query Prädikate nur unter einer non-CDB Instanz appliziert werden, d.h. in einer Multitenant Konfiguration werden die PERIOD FOR Klauseln und die DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME Calls ignoriert.

Eine weitere Limitation ist, dass Oracle 12c noch keine Unterstützung für temporale Joins und temporale Aggregationen bietet.

Tri-Temporales Datenmodell - Beispiel
Das nachfolgende Datenmodell basiert auf dem EMP/DEPT Modell im Schema SCOTT und implementiert für die Tabelle EMPV die Transaktionszeit (TT) mit Flashback Data Archive und die Gültigkeitszeit (VT) und Entscheidungszeit (DT) mit Temporal Validity.

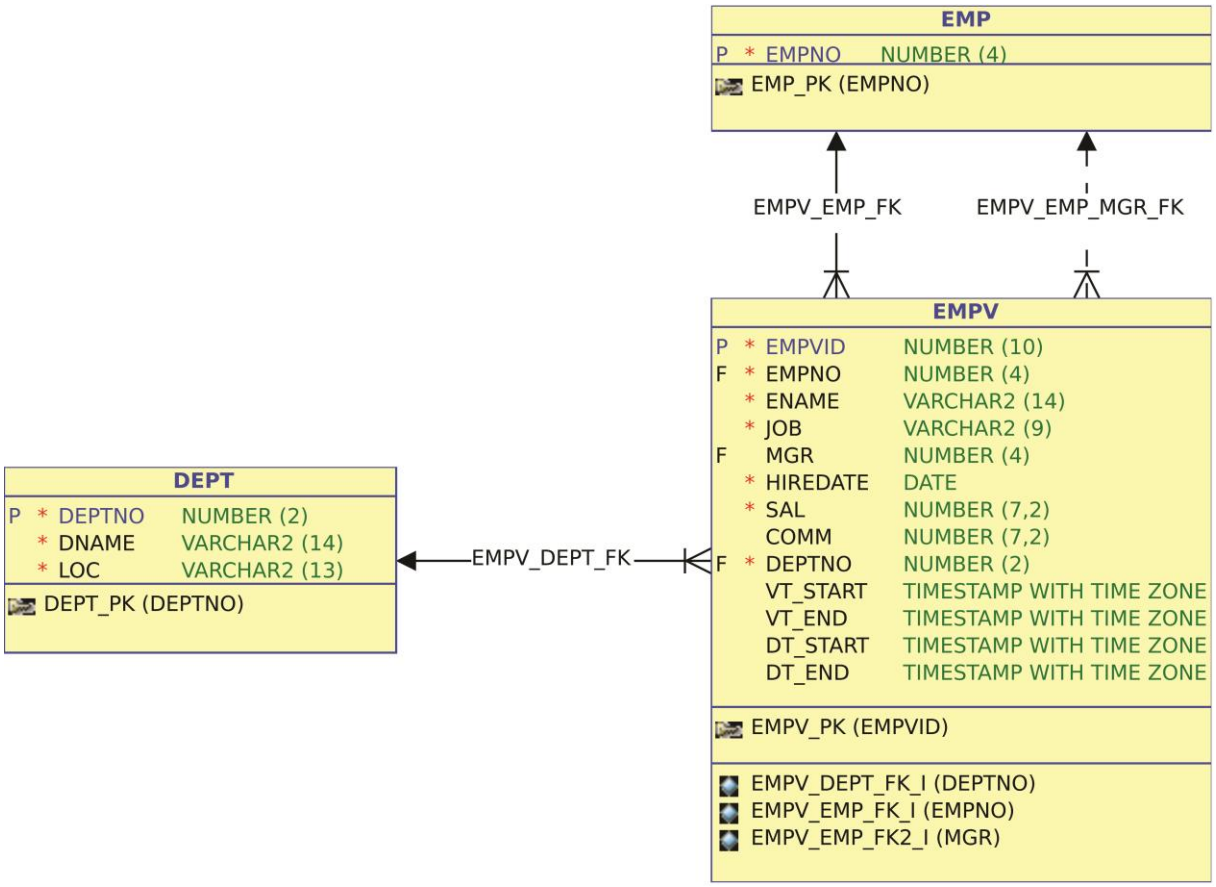

Abb. 4: Tri-Temporales Datenmodell mit FBDA für TT

Die Tabelle EMP ist auf den Primärschlüssel reduziert, so dass der Fremdschlüssel EMPV_EMP_MGR_FK angelegt und aktiviert werden kann. Folgende sechs Ereignisse werden in diesem Modell abgebildet:

No	Transaction Time (TT)	Valid Time (VT)	Decision Time (DT)	Action
#1	1			Initial load from SCOTT.EMP table
#2	2	1990-01-01		Change name from SCOTT to Scott
#3	3	1991-04-01		Scott leaves the company
#4	4	1991-10-01		Scott rejoins
#5	5	1989-01-01		Change job from ANALYST TO Analyst
#6	6	2014-01-01	2013-03-24	Change job to Manager and double salary

Abb. 5: EMP/EMPV Ereignisse

Nach der Verarbeitung des Ereignisses #6 können alle Perioden des Mitarbeiters 7788 (Scott) in der Tabelle EMPV wie folgt abgefragt werden. Die Transaktionszeit wird durch die System Change Number SCN repräsentiert.

```
SQL> SELECT dense_rank() OVER(ORDER BY versions_startscn) event_no, empno, ename, job,
  2         sal, versions_startscn tt_start, versions_endscn tt_end,
  3         to_char(vt_start,'YYYY-MM-DD') vt_start, to_char(vt_end,'YYYY-MM-DD') vt_end,
  4         to_CHAR(dt_start,'YYYY-MM-DD') dt_start, to_char(dt_end,'YYYY-MM-DD') dt_end
  5    FROM empv VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
  6   WHERE empno = 7788 AND versions_operation IN ('I','U')
  7   ORDER BY tt_start, vt_start NULLS FIRST, dt_start NULLS FIRST;

# EMPNO ENAME JOB       SAL TT_START   TT_END VT_START   VT_END     DT_START   DT_END
-- ----- ----- ------- ----- -------- -------- ---------- ---------- ---------- ----------
 1  7788 SCOTT ANALYST  3000  2366310  2366356
 2  7788 SCOTT ANALYST  3000  2366356  2366559            1990-01-01
 2  7788 Scott ANALYST  3000  2366356  2366408 1990-01-01
 3  7788 Scott ANALYST  3000  2366408  2366559 1990-01-01 1991-04-01
 4  7788 Scott ANALYST  3000  2366424  2366559 1991-10-01
 5  7788 SCOTT ANALYST  3000  2366559                     1989-01-01
 5  7788 SCOTT Analyst  3000  2366559           1989-01-01 1990-01-01
 5  7788 Scott Analyst  3000  2366559           1990-01-01 1991-04-01
 5  7788 Scott Analyst  3000  2366559  2366670  1991-10-01
 6  7788 Scott Analyst  3000  2366670           1991-10-01                      2013-03-24
 6  7788 Scott Analyst  3000  2366670           1991-10-01 2014-01-01 2013-03-24
 6  7788 Scott Manager  6000  2366670           2014-01-01            2013-03-24
```

Das obigen Resultat zeigt, dass vier Datensätzen durch das Ereignis #5 zum Transaktionszeitpunkt 2366559 verändert worden sind. Dadurch wird deutlich, dass DML Operationen in einem Multi-temporalen Modell nicht trivial sind. Umso mehr wird eine Unterstützung in diesem Bereich vermisst.

Die nachfolgende Query schränkt den obigen Datenbestand auf der Transaktionzeit (Sysdate=Default), Gültigkeitszeit (01.01.2014) und Entscheidungszeit (01.04.2013) ein, so wird das Ergebnis auf maximal einen Datensatz reduziert.

```
SQL> SELECT empno, ename, job, sal,
  2         to_char(vt_start,'YYYY-MM-DD') AS vt_start,
  3         to_char(vt_end,'YYYY-MM-DD') AS vt_end,
  4         to_CHAR(dt_start,'YYYY-MM-DD') AS dt_start,
  5         to_char(dt_end,'YYYY-MM-DD') AS dt_end
  6    FROM empv AS OF period FOR dt DATE '2013-04-01'
  7   WHERE empno = 7788 AND
  8         (vt_start <= DATE '2014-01-01' OR vt_start IS NULL) AND
  9         (vt_end > DATE '2014-01-01' OR vt_end IS NULL)
 10   ORDER BY vt_start NULLS FIRST, dt_start NULLS FIRST;

EMPNO ENAME JOB       SAL VT_START   VT_END     DT_START   DT_END
----- ----- ------- ----- ---------- ---------- ---------- ----------
 7788 Scott Manager  6000 2014-01-01            2013-03-24
```

Die Abfrage von multi-temporalen Daten ist relativ einfach, sofern für alle Zeitdimensionen einer Tabelle nur ein einzelner Zeitpunkt betrachtet wird. Die AS OF PERIOD Klausel (für DT) vereinfacht die Abfrage zwar, aber die Komplexität einer traditionellen WHERE Condition (für VT) ist hier nicht viel höher.

Fazit

Die Unterstützung für temporale Datenhaltung in Oracle 12c basiert auf fundierten, ausgereiften Konzepten, aber die Implementation ist unvollständig. Wir vermissen vor allem ein temporales DML API, temporale Integrity Constraints, temporale Joins und temporale Aggregationen. Wir empfehlen in bestehenden Modellen Oracle's Semantik für Perioden zu verwenden (halboffene Intervalle), um eine spätere Migration zu vereinfachen.

In der realen Welt verwenden wir viele temporale Zeitdimensionen bewusst oder unbewusst gleichzeitig. In einem Datenmodell hingegen erhöht jede temporale Zeitdimension die Komplexität signifikant. Datenmodelle sind Vereinfachungen der realen Welt, basierend auf Anforderungen und einem limitierten Budget. Es ist nicht zu empfehlen Bitemporalität oder gar Multi-Temporalität als durchgängiges Design-Muster anzuwenden, um quasi auf künftige Anforderungen besser vorbereitet zu sein. Im Gegenteil, die Gründe für jede temporale Zeitdimension sind pro Entität zu eruieren und festzuhalten, so dass temporale Zeitdimensionen bewusst und nicht unnötigerweise modelliert werden.

Oracle's Flashback Data Archive ist eine gute, transparente und seit Oracle 11.2.0.4 auch eine kostenfreie Option zur Umsetzung der Anforderungen betreffend der Transaktionszeit. Für andere Zeitdimensionen wie Gültigkeitszeit und Entscheidungszeit empfehlen wir Werkzeuge oder Frameworks für DML Operationen einzusetzen. Trivadis verfügt über eine langjährige Erfahrung mit bitemporalen Daten und dem Einsatz solcher Frameworks. Gerne unterstützen wir Sie bei Ihren Vorhaben.

Kontaktadresse:
Philipp Salvisberg
Trivadis AG
Europastrasse 5
CH-8152 Glattbrugg (Zürich)

Telefon:	+41-58-459 55 55
Fax:	+41-58-459 55 95
E-Mail	philipp.salvisberg@trivadis.com
Internet:	www.trivadis.com