Temporary Tables

Temporary Tables sind Tabellen, die temporäre Daten enthalten z. B. für Sortieren von Zwischenresultaten, die für Speichersortierung zu groß sind. Die zwei Arten von TT's sind **"created temporary tables"** und **"declared temporary tables".**

Wozu sollte man bei DB2 also Temp-Tables verwenden ?

Zum **Speichern von Resultaten aus Queries**, die im späteren Ablauf mehrfach benötigt werden. Dies ist insbesondere bei komplexen und aufwändigen Queries sinnvoll.

Oder, wenn ein **Ergebnis mehrfach innerhalb eines Programms** an den User zurückgegeben werden soll. Einmal gesucht, kann in folgenden Anforderungen die benötigte Information aus der Temp Table gelesen werden.

„Temporary tables" können auch dazu benutzt werden, **"non-relational data" mit SQL zu bearbeiten**: Zum Beispiel kann man eine "**global temporary table"** erzeugen, um sie mit IMS-Daten (oder Daten aus anderen „non-relational" Datenquellen) zu füllen und sie im Anschluss mit SQL Statements zu bearbeiten.

Ein weiterer Grund für die IBM "temporary tables" zu unterstützen, war es, die **Migration von anderen relationalen Datenbanken auf DB2** einfacher zu gestalten. Microsoft SQL Server und Oracle unterstützen "temporary tables" ebenfalls.

"Created Temporary Tables"

Eine "created temporary table" existiert solange wie ein Prozess sie nutzt. „Temporary tables" werden angelegt über das **CREATE GLOBAL TEMPORARY TABLE Statement**. Einmal erzeugt, wird das "table-schema" im DB2 Systemkatalog (SYSIBM.SYSTABLES) wie jede andere Tabelle gespeichert. Dabei wird die Spalte TYPE auf das Kennzeichen '**G**' gesetzt.

Es ist wichtig zu wissen, dass eine "created global temporary table" über ein DDL CREATE Statement angelegt werden muss, bevor sie von einem Programm benutzt werden kann.

Eine **"created temporary table"** wird initialisiert, wenn sie in einem OPEN, SELECT INTO, INSERT, oder DELETE Statement angesprochen wird. **Jeder „application process", der die "temporary table" nutzt, erzeugt eine eigene "instance"**. Bei der Nutzung von TTs sollte folgendes bedacht werden:

- Weil **nicht "persistent"** gibt es bestimmte typische "database operations" inklusive "locking", "logging", und "recovery" nicht für "created temporary tables".
- **Indexe** können für "created temporary tables" **nicht** angelegt werden
- **Constraints** können für "created temporary tables" **nicht** angelegt werden
- **NULL** ist der einzige „default"-Wert, der bei CTT für Spalten erlaubt wird
- „Created temporary tables" können nicht **von DB2 Utilities** verarbeitet werden
- "Created temporary tables" können **nicht mit UPDATE** verändert werden
- Beim **DELETE** auf CTT müssen alle "rows" gelöscht werden
- **"Views"** können über CTTs **jederzeit** angelegt werden.Die WITH CHECK OPTION ist jedoch nicht erlaubt

Zur Verwaltung der “created temporary tables” werden “Work file data sets” eingesetzt. Die “work database” (DSNDB07) wird als Speicher auch für die Verarbeitung von SQL-Statements – nicht nur für “created temporary tables” verwendet.

Wird eine “temporary work file” über INSERT gefüllt, so kann kein anderer Prozess dieselbe “work file” nutzen bis die CTT wieder freigegeben wird. Das geschieht, wenn die Applikation COMMIT oder ROLLBACK durchführt, bzw. wenn ein DEALLOCATE passiert, abhängig von der RELEASE Option, die beim BIND gesetzt wurde. So scheint es eine gute Idee zu sein, **für die „work files” eigene Bufferpools** zuzuweisen, zusätzlich dazu aber die CTTs von den anderen „work files” zu trennen.

Declared Temporary Tables

Mit der Version 7 von DB2, bietet die IBM nun sogenannte **„declared temporary tables“ (DTT)** an.

Dieser neue Typ von “temporary tables” unterscheidet sich grundlegend von den anderen “created temporary table” Typen, insbesondere aufgrund der Nutzungsbeschränkungen. Ein wichtiger Unterschied ist die **Verwendung eines DECLARE Statements in einem Applikationsprogramm** – und nicht eines DDL CREATE Statement. Da DTTs nicht „persistent” sind besitzen sie auch **keine Beschreibung im DB2-Katalog**.

Zusätzlich dazu bieten “declared temporary tables” besondere “features” und Funktionalität, die von “created temporary tables” nicht geleistet werden:

- **“Declared temporary tables”** können indiziert sein und mit CHECK “constraints” versehen werden
- **UPDATE Statements und DELETE** sind auf “declared temporary tables” möglich
- **Die “columns” einer DTT können implizit definiert** und über SELECTs zugegriffen werden

Um eine “instance” für eine “declared temporary table” zu erzeugen, muss man ein **DECLARE GLOBAL TEMPORARY TABLE Statement** innerhalb eines Programms absetzen. Diese “instance” ist nur dem Prozess bekannt, der das DECLARE Statement abgesetzt hat. **Mehrere konkurrierende Programme können dieselbe “declared temporary table” Namen** verwenden, da jedes Programm eine eigene Kopie der “temporary table” besitzt.

Aber DTTs verlangen mehr Vorarbeiten als CTTs. Es muss **eine “temporary database” und zugehörige “table spaces”** verfügbar gemacht werden. Erreicht wird dies indem die Klausel AS TEMP beim CREATE DATABASE Statement angegeben wird.

Dann sind “segmented table spaces” in dieser “temporary database” zur Verfügung zu stellen. Es ist nur EINE “temporary database” für DTTs pro DB2 Subsystem erlaubt.

Die User von “temporary tables” müssen mit USE auf die TS authorisiert sein.

Beispiel:

```
DECLARE GLOBAL TEMPORARY TABLE TEMP_EMP
  (EMPNO      CHAR(6)     NOT NULL,
   FIRSTNME   VARCHAR(12) NOT NULL,
   MIDINIT    CHAR(1)     NOT NULL,
   LASTNAME   VARCHAR(15) NOT NULL,
   WORKDEPT   CHAR(3),
   PHONENO    CHAR(4)
  );
```

Dies erzeugt eine **“declared temporary table” TEMP_EMP.** Die LIKE Klausel ist ebenfalls erlaubt, um eine “temporary table” nach einem existierenden Schema zu definieren. Es ist ausserdem möglich die INCLUDING IDENTITY COLUMN ATTRIBUTES Klausel zu verwenden.

Beispiel:

```
DECLARE GLOBAL TEMPORARY TABLE TEMP_PROJ
 LIKE DSN8710.PROJ
 INCLUDING IDENTITY
 ON COMMIT PRESERVE ROWS;
```

Die **ON COMMIT PRESERVE ROWS** Klausel spezifiziert was DB2 mit den Daten in der DTT tun soll, wenn das Programm ein COMMIT Statement setzt. Es gibt zwei Optionen: PRESERVE oder DELETE “rows”. **“Default” ist die ON COMMIT DELETE ROWS** option.

“Scrollable cursors” beispielsweise benötigen “declared temporary tables”.

```
CREATE DATABASE TEMPDB AS TEMP;

CREATE TABLESPACE TEMPTS
 IN TEMPDB
 SEGSIZE 4
 BUFFERPOOL BP7;
```

Declared Temporary Tables und Performance der AP

Bei **“declared temporary tables”** werden keine Einträge in den Systemkatalog getätigt, so werden “contentions” auf dem Katalog vermieden. Im Unterschied zu normalen Tabellen werden “declared temporary tables” oder ihre “rows” **nicht “gelocked”** und bei Angabe der **NOT LOGGED Parameter** auch nicht **“gelogged“.**

Immer wenn eine Anwendung grosse Datenmengen verarbeitet, die nach der Verarbeitung wieder freigegeben werden, so sollten “declared temporary tables“ anstatt normaler DB2-Tabellen verwendet werden.

Entwickelt man Applikationen für **“concurrent users”**, so sind “declared temporary tables” von Vorteil. Bei “declared temporary tables” gibt es keine Namenskollisionen. Man definiert EINE “declared temporary table” und DB2 garantiert, dass jede “instance” seine eigene Tabelle nutzen kann.

Beispiel:

```
DECLARE GLOBAL TEMPORARY TABLE TT1

SELECT column1 FROM SESSION.TT1;
```

RUNSTATS gegen “declared temporary tables” und ihre Indizes ist immer empfehlenswert.

Ein **Nachteil bei DTTs** besteht darin, dass, wenn der **Thread inaktiv** ist, er zwischenzeitlich anderweitig genutzt werden kann. Dies kann sich u.U. nachteilig auf den Durchsatz auswirken. Die Grenze für Threads ist zwar hoch, kann aber bei großen Installationen mit vielen DTTs durchaus erreicht werden.

Ein Nachteil ist auch das **Handling**, weil man nur schwer **Indizes mit korrekten Runstats-Statistiken** verwenden kann.