„IDENTITY table clause“-Feature und Apex

Alain Lacour und Arnaud Berbier, dbi-services

Der Artikel beschreibt, wie in Apex Primärschlüssel auf einer „IDENTITY“-Spalte abgebildet werden können, sodass eine harmonische und performante Implementierung mit Oracle 12*c* gelingt.

Die neue Datenbank bietet für die Tabellenerstellung die neue Klausel „IDENTITY“ für eine nummerische Spalte. Diese generiert bei Bedarf für jeden Spalten-Eintrag eine Sequenznummer. Der exakte Bedarf wird mit einem Klausel-Parameter deklariert. Laut Dokumentation gibt es folgende Parameterwerte:

- ALWAYS (Default)
- BY DEFAULT
- BY DEFAULT ON NULL

Je nach Klausel-Parameter werden DML-Operationen unterschiedlich behandelt. Bemerkenswerterweise betrifft diese neue Deklaration der Vorgabewerte teilweise auch „UPDATE-DML“. Dazu ein Beispiel mit der Datenbank-Version 12.1.0.1.0 und Apex 4.2.0.00.27, das bereits in der Datenbank-Software enthalten ist. „ALWAYS“ verhindert, dass ein Wert manuell eingetragen wird. Während der „INSERT“- oder „UPDATE“-Vorgänge darf diese „IDENTITY“-Spalte nicht adressiert werden. Bei Nichtbeachtung wird eine DML-Exception generiert (siehe Listing 1).

```
CREATE TABLE md_country
(
  country_id   NUMBER GENERATED ALWAYS AS IDENTITY,
  country_code CHAR (3 BYTE) NOT NULL,
  country_name VARCHAR2 (200 BYTE) NOT NULL
);
```

Listing 1

```
CREATE TABLE md_country
(
  country_id   NUMBER GENERATED BY DEFAULT AS IDENTITY,
  country_code CHAR (3 BYTE) NOT NULL,
  country_name VARCHAR2 (200 BYTE) NOT NULL
);
```

Listing 2

```
CREATE TABLE md_country
(
  country_id   NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
  country_code CHAR (3 BYTE) NOT NULL,
  country_name VARCHAR2 (200 BYTE) NOT NULL
);
```

Listing 3

„BY DEFAULT“ bietet die Möglichkeit, einen manuellen Wert anzugeben – außer „NULL“. So kann die Spalte während „INSERT“- oder „UPDATE“-Operationen manipuliert werden, solange ihr neuer Wert nicht „NULL“ ist (siehe Listing 2).

„BY DEFAULT ON NULL“ bietet die Möglichkeit, einen manuellen Wert

anzugeben – inklusive „NULL“. Sollte die DML-Anweisung den „NULL“-Wert für die „IDENTITY“-Spalte vorsehen, erzwingt die Klausel „BY DEFAULT ON NULL“ einen Wert aus der Sequenz. So kann die Spalte bei „INSERT“- oder „UPDATE“-Operationen adressiert werden, sogar mit „NULL“ als Wert in der DML-Anweisung (siehe Listing 3). Das ist sehr nützlich im Apex-Framework, da Apex für Primary Keys vorzugsweise Nummern verwendet.

Die Apex-Version, die mit Oracle 12*c* geliefert wird, unterstützt bereits das neue Feature, sodass bereits im SQL-Workshop beim Aufsetzen einer Tabelle die Primary Keys entsprechend deklariert werden können. Dafür wurden ein neues Spalten-Attribut im Wizard „Create Table – Columns“ sowie eine neue Auswahlmöglichkeit auf der „Create Table – Primary Key“-Seite hinzugefügt, was im Folgenden beschrieben wird. Zuerst wird die Spalte als „IDENTITY“ deklariert (siehe Abbildung 1).

Hinweis: Der Identity-Parameter kann nur gesetzt werden, wenn der Datentyp „NUMBER“ gewählt ist. Dann wird die Primary-Key-Steuerung auf „Identity Column“ gesetzt (siehe Abbildung 2).

Wenn man am Ende der Tabellendefinition die verschiedenen Objekte mittels SQL-Workshop betrachtet (siehe Abbildung 3), entdeckt man Folgendes:

- *Tabelle*
  Die über eine „IDENTITY“-Klausel definierte Spalte besitzt einen Default-Wert, ermittelt aus einer Sequenz, was auf einem anderen Weg nicht erlaubt wäre. So kann eine solche Spalte erkannt werden, obwohl es keine explizite „IDENTITY“-Indikatoren gibt.
- *Sequenz*
  Bei einer Untersuchung der Sequenz fällt der merkwürdige Name auf. Anzumerken ist zudem, dass die Sequenz einen Default-Cache bekommen hat, wodurch Lücken bei Vergabe von „COUNTRY_ID“-Nummern entstehen können, wenn von gleichzeitigen Sessions nur vereinzelte Einträge gemacht werden. Dieses Verhalten kann mit fortgeschrittener Syntax der Tabellen-Definition laut Oracle-Dokumentation genauer spezifiziert werden (siehe Abbildungen 4 und 5).

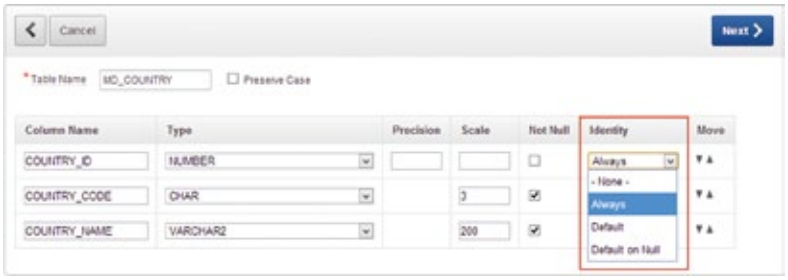

Abbildung 1: Create Table Columns

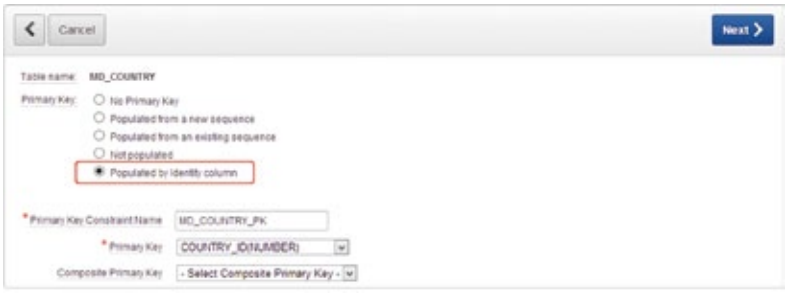

Abbildung 2: Create Table Primary Key

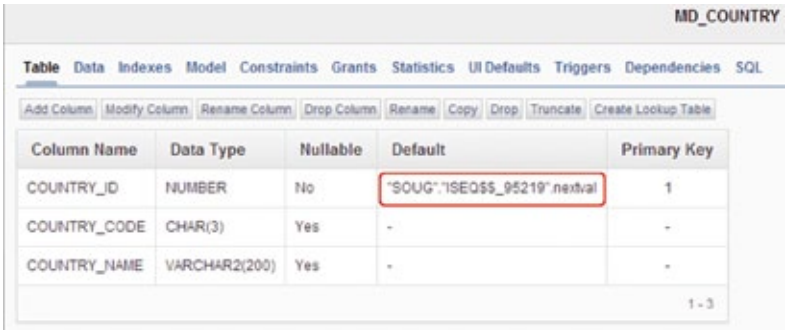

Abbildung 3: Object Browser Table Definition

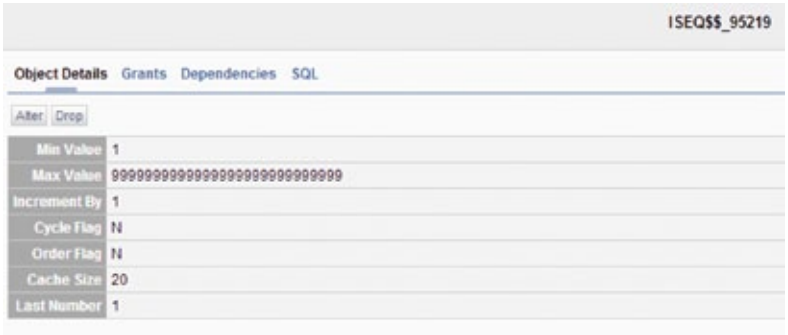

Abbildung 4: Object Browser Sequence Details

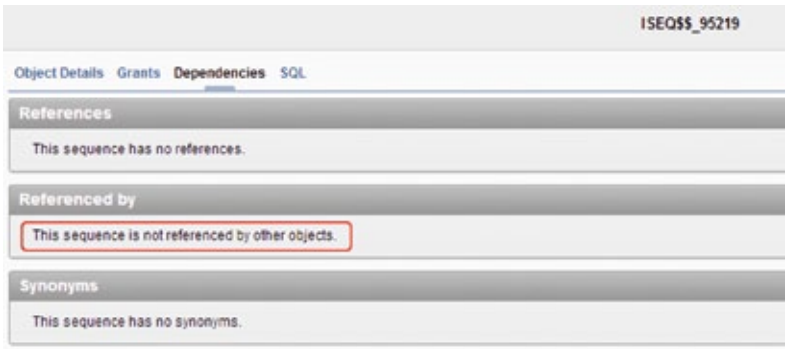

Abbildung 5: Object Browser Sequence Dependencies

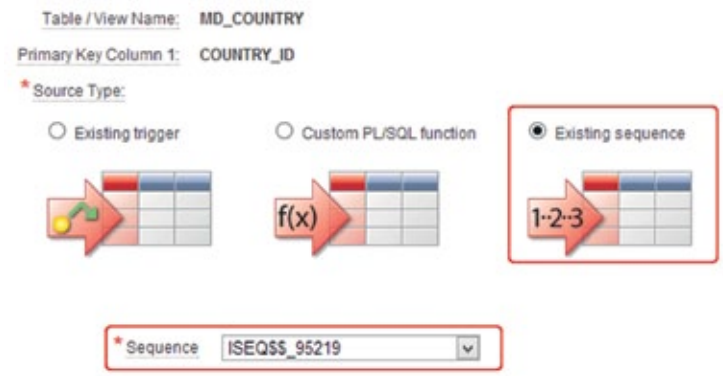

Abbildung 6: Create Page Primary Key

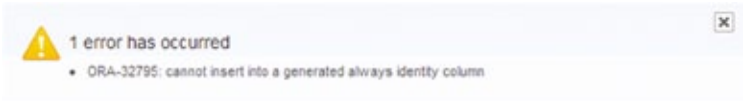

Abbildung 7: Error message

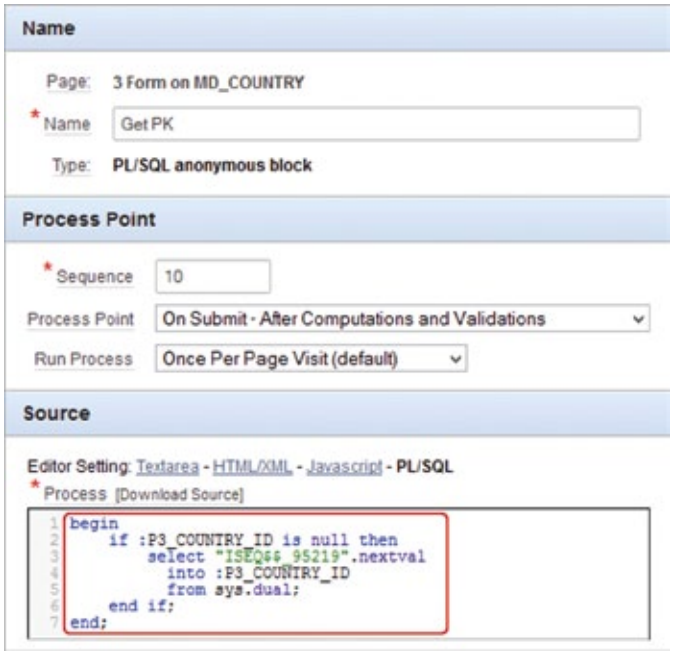

Abbildung 8: Edit Page Process

Hervorzuheben ist die verwirrende Tatsache, dass, obwohl die Sequenz durch die Verwendung einer „IDENTITY“-Klausel entstanden ist, die Tabelle nicht in der Referenzliste erscheint.

Behandlung der „IDENTIY“ als Primary Key in einem Formular

Damit das Zusammenspiel zwischen dem neuen Feature und Apex getestet werden kann, wurde eine Datenbank-Applikation erstellt. Darin verwaltet ein Formular die Daten der „MD_COUNTRY“-Tabelle.

Fall 1, mit „ALWAYS“-Parameter: Bei der Wahl der Primary-Key-Quelle (während der Erstellung des Formulars) wurde manuell „Existing sequence“ gewählt. Auch die Sequenz, die zuvor beim Aufsetzen der Tabelle generiert worden ist, wurde Apex explizit in der Drop-Down-Liste „Sequence“ bekanntgegeben (siehe Abbildung 6).

Beim Anlegen neuer Datensätze mithilfe dieses Formulars kommt es jedoch zu einer Fehlermeldung (siehe Abbildung 7). Der fehlschlagende SQL-Code liegt in der „Get PK“-Prozess-Anweisung, die von Apex für den Formularbereich „Page Processing“ generiert wurde (siehe Abbildung 8).

Man erkennt im Quellcode, dass für die Spalte ein Wert gesetzt wird. Eine „IDENTITY ALWAYS“-Spalte verhindert so etwas bekanntlich und wirft eine DML-Exception. Eine Daten-Änderung („update“, „delete“) kann weiterhin ohne Fehlermeldung durchgeführt werden. Wenn man hingegen „Existing Trigger“ statt „Existing sequence“ wählt, werden die Daten auch beim Erstellen richtig behandelt. Die „Get PK“-Prozessanweisung wird bei dieser Variante nicht generiert (siehe Abbildung 9).

Obwohl kein Trigger existiert, behandelt Apex die Tabelle so, als ob es einen gäbe. Somit kümmert sich Apex gar nicht um die Verwaltung der Primary-Key-Spalte und überlässt das der Datenbank. Glücklicherweise stört es Apex nicht, dass es keinen Trigger gibt.

Im Fall 2, mit „BY DEFAULT“-Parameter, macht es keinen Unterschied, ob man „Existing Trigger“ oder „Existing sequence“ während der Erstellung des Formulars wählt, denn hier ist die Angabe eines Werts für die „IDENTITY“-Spalte erlaubt, solange von „NULL“ verschiedene Werte angegeben sind. So hat die „Get PK“-Prozessanweisung keinen Einfluss auf die korrekte Arbeitsweise der „IDENTITY“-Spalte, solange sie nicht geändert wird, um den „NULL“-Wert zu verwenden.

Im Fall 3, mit „BY DEFAULT ON NULL“-Parameter, ist es ebenfalls nebensächlich, ob man „Existing Trigger“ oder „Existing sequence“ verwendet. Auch hier akzeptiert die „IDENTITY“-Spalte Werte einschließlich „NULL“. Die „Get PK“-Prozessanweisung findet an dieser Stelle keinen Einsatz.

Anmerkung: Die dritte Wahl, „Custom PL/SQL function“, sollte für alle Möglichkeiten und die „IDENTITY“-Klausel-Parameter-Werte sorgfältig erwogen werden, bevor sie aufgesetzt wird.

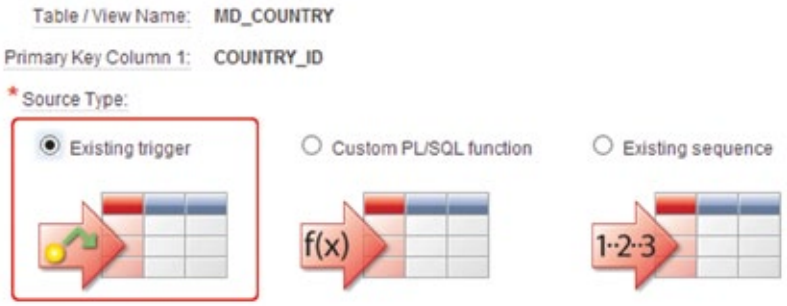

Abbildung 9: Create Page Primary Key

Anzahl der eingefügten Datensätze	Zeitverbrauch der Identity-Variante (s)	Zeitverbrauch der Trigger-Variante (s)
100	0.0071	0.0232
1000	0.0486	0.2043
10000	0.4737	2.1988
100000	5.0842	21.9390
1000000	53.8582	217.648

Tabelle 1: Performance-Vergleich

Betrachtung der Performance

Eine Methode für die Handhabung von ID-Spalten ist, dass man ihre Werte mithilfe eines „Before Insert"-Triggers automatisch generieren lässt. Dieser verwendet seinerseits ein Sequenzobjekt. Das wird mit der neuen 12*c*-IDENTITY-Funktionalität verglichen und hinsichtlich der Performance untersucht. Es wurden Ergebnisse für die vierfache Geschwindigkeit der „IDENTITY"-Generierung im Vergleich zur Trigger-Lösung erzielt, unabhängig von der Anzahl der Datensätze (siehe Tabelle 1).

Fazit

Da für die Wahl der Primary-Key-Quelle die „Existing Trigger"-Option in jedem Fall funktioniert, scheint sie die bessere Wahl zu sein, sobald eine „IDENTITY"-Spalte eine Verwendung findet. Hinzu kommen die bemerkenswerten Performance-Vorteile der „IDENTITY"-Spalte, die für ihre Verwendung als ID-Spalte in Oracle 12*c* sprechen.

Alain Lacour
alain.lacour@
dbi-services.com

Arnaud Berbier
arnaud.berbier@
dbi-services.com