Namenskonventionen – Sinn oder Unsinn?

Autor: Michael A. Istinger, EDV-Beratung Istinger

DOAG *News* Q2_2005

Bei jedem deutschen Konzern fällt mir früher oder später ein Dokument in die Hände, das sich bedeutungsschwer „Richtlinie für Datenbank-Entwicklungen“, „Namenskonventionen für Oracle-Datenbanken“, „Infrastruktur-Richtlinien für Datenbanken“ oder so ähnlich nennt. Die Namen mögen unterschiedlich sein, die Inhalte gleichen sich (leider!) wie ein Ei dem anderen. Es geht immer um den Versuch, den Typ von Datenbank-Objekten in den Namen derselben als Prä- oder Postfix aufzunehmen. In dem mir gerade auf dem Tisch liegenden Dokument liest sich das auszugsweise so:

```
Präfixe für Datenbank-Objekte

      SG_   = Snapshotgruppen
      SN_   = Snapshots
      IX_   = Indexes
      ST_   = Snapshots auf prebuilt
              Tables
      MV_   = Materialized View
      TB_   = Tables
      PT_   = Partition Tabellen
      VW_   = Views
      PX_   = Partition Indexes
      IOT_  = Index Organized Tables

Constraints

Primary Key Constraint
<Tabellenbezeichnung>_PK

Foreign Key Constraint
<Tabellenbezeichnung_FKi, i ist eine
laufende Nummer.
```

Ich habe noch nie verstanden, warum Konzerne diese Art von Standard-Dokumenten publizieren, die – wie die Praxis zeigt – nicht funktionieren können. Warum?

Namenskonventionen verteuern die Software-Entwicklung

Betrachten wir das eingang erwähnte Beispiel. Demnach müssen Index-Namen mit dem Präfix IX_ beginnen, Primary Key Constraints mit dem Tabellennamen beginnen und mit dem Postfix _PK enden. Diese scheinbar harmlosen Forderungen führen dazu, dass es nicht möglich ist, Primary Key Constraints – wie es häufig vorkommt – einfach beim entsprechenden **`CREATE TABLE`**-Statement mit aufzunehmen. Es muss viel mehr:

- Zunächst die Tabelle ohne Primärschlüssel erzeugt werden.
- Anschließend ein Index angelegt werden, dessen Name den Namenskonventionen entspricht.
- Erst im dritten Schritt kann mittels eines **`ALTER TABLE`** Statements die Tabelle um ein den Namenskonventionen entsprechendes Primärschlüssel-Constraint erweitert werden, wobei dieses Constraint dann intern den im zweiten Schritt erzeugten Index verwendet.

Die Folgen dieser Namenskonventionen sind also längere Scripts zum Erzeugen der benötigten Datenbank-Objekte sowie erhöhter Test- und Pflegeaufwand.

Namenskonventionen sind nicht eindeutig

Der gezeigte Ausschnitt aus den Namenskonventionen bietet für Materialized-Views drei unterschiedliche Namenspräfixe an: **`MV_`**, **`SN_`** und **`ST_`**. Dabei fällt zunächst auf, dass innerhalb der Oracle-Welt die Begriffe Snapshot und Materialized-View völlig synonym verwendet werden. Für den Entwickler ist es absolut nicht ersichtlich, welches Namenspräfix hier anzuwenden ist, entsprechend sind auch kreative Lösungen wie „Wir verwenden diese beiden Kürzel eben abwechselnd“ durchaus denkbar und durch die Namenskonvention gedeckt.

Das Namenspräfix **`ST_`** ist ein besonderer Liebling des Autors. Ob ein Snapshot über einer vorher existierenden Tabelle aufgebaut werden soll oder nicht, hat zu genau zwei Zeitpunkten Relevanz: Beim Anlegen des Snapshots und beim Löschen. Zwischen diesen beiden Zeitpunkten ist ein Snapshot, der über einer vorher existierenden Tabelle aufgebaut worden ist, einfach ein Snapshot, der sich genau so verhält wie jeder andere Snapshot auch. Warum hierfür eine eigenen Namenskonvention vorgesehen wurde, ist völlig unklar.

Man muss sich an dieser Stelle auch vor Augen halten, dass die Entscheidung, ob ein Snapshot über einer bereits existierenden Tabelle aufgebaut werden soll oder nicht, meist erst sehr spät im Lebenszyklus eines Projekts auftritt. Typischerweise möchte das Projekt nur einen Snapshot haben, es wird also gegen einen normalen Snapshot entwickelt und getestet. Meist stellt sich erst kurz vor oder während der Produktionseinführung heraus, dass das initiale Laden des Snapshots über das Netzwerk aus externen Überlegungen nicht möglich ist, und erst zu diesem Zeitpunkt – zu dem die Entwicklung und alle Tests bereits abgeschlossen sind – wird dann beschlossen, den Snapshot über einer schon existierenden Tabelle aufzubauen. Dann fordern aber die Namenskonventionen, dass der Name des Snapshots geändert werden muss – und das erfordert wieder Änderungen an der Software, erneutes Testen, neue Rollout-Pläne usw. – und das alles nur im der Namenskonvention Genüge zu tun!.

Namenskonventionen sind nicht vollständig

Ich habe noch nie eine Namenskonvention gesehen, die tatsächlich alle, in einer neueren Oracle-Version möglichen Datenbank-Objekt-Typen enthält.
Mal fehlen Bitmap Indexes, mal Queue-Tables.
Es scheint wohl so zu sein, dass die Liste immer genau die Objekt-Typen enthält, die gerade auf einer Datenbank, die dem Verfasser der Namenskonventionen bekannt ist, vorkommen.

Verwendet nun ein Projekt einen Objekt-Typ, der in den lokal üblichen Namenskonventionen nicht berücksichtigt wird, so muss ein formaler Prozess durchlaufen werden, der nicht selten Monate dauert, um diesen einen Objekt-Typ in einer neuen Version der Namenskonventionen berücksichtigt zu bekommen.

Ich kenne mehr als ein Projekt, in dem – nach früheren leidvollen Erfahrungen – vorgezogen wird, diesen Prozess gar nicht erst einzuleiten.

Namenskonventionen verteuern die Wartung und Weiterentwicklung von existierenden Projekten

Es ist nicht unüblich, dass sich im Laufe der Lebenszeit eines Projekts der Typ von Datenbank-Objekten ändert: Die Tabelle aus Release 1.0 des Projekts soll in Release 2.5 durch einen Snapshot ersetzt werden. Tabellenstrukturen ändern sich, eine Tabelle soll durch einen View ersetzt werden, usw. Die Dynamik in diesem Bereich ist hoch. Üblicherweise sind solche Änderungen kein Problem, und können mit den Standard-Mechanismen einer relationalen Datenbank einfach geregelt werden. Die Änderungen können in der Regel auch so vorgenommen werden, dass gegenüber der Anwendung völlig transparent sind.

Genau diese Transparenz machen aber nun die Namenskonventionen unmöglich. Die Konventionen fordern ja, dass sich mit dem Typ des Objekts auch der Name des Objekts ändern muss – und damit ist diese Änderung eben nicht mehr transparent für die auf der Datenbank operierenden Applikationen! Die Namenskonventionen führen dazu, dass notwendige Änderungen an der Datenbank-Struktur immer zu (aufwändigen!) Änderungen am Code der entsprechenden Applikationen führen, und auch das Rollout von Datenbank-Strukturen auf das Engste mit dem Rollout des entsprechend geänderten Applikationscodes verknüpft durchgeführt werden muss. Der Praktiker weiß, dass durch das skizzierte Vorgehen Mehrkosten in kaum kalkulierbarer Höhe entstehen.

In der Praxis finden sich deshalb bei solchen Datenbank-Strukturänderungen immer wieder Verletzungen der Namenskonventionen, die bewusst deshalb in Kauf genommen wurden, weil die Mehrkosten für die Einhaltung der Konvention höher als das vorhandene Budget waren.

Namenskonventionen behindern den Einsatz von „Standard-Komponenten“

Standard-Komponenten wie z.B. PL/VISION oder QNXO folgen den Namenskonventionen nicht, sondern verwenden ihre eigenen. Will nun ein Projekt solche Standard-Komponenten einsetzen, so

ist das unter Einhaltung der Konvention nicht möglich. Die übliche Vorgehensweise an dieser Stelle ist dann, für die Objekte solche Komponenten einen Ausnahmestatus definieren zu lassen – also, mit anderen Worten, die Namenskonvention nicht zu befolgen.

Es gibt in Praxis keine Alternative zu dieser Vorgehensweise. Das Anpassen von Standard-Kompo-nenten an die beim entsprechenden Konzern gültigen Namenskonventionen erfordert einen so großen Aufwand, und ist mit so hohen Risiken verbunden, dass sie nicht ernsthaft in Erwägung gezogen werden können. Durch den Einsatz von Standard-Komponenten soll schließlich der Umfang des individuell zu erstellenden Codes verringert werden, nicht erhöht!

Ähnliche Aussagen gelten auch für Datenbank-Überwachungstools. BMC Patrol, Oracle Enterprise Manager, Quest Central for Oracle … keines dieser Produkte folgt dem lokalen Namenskonventionen und dennoch werden sie eingesetzt!

Schlussbemerkung

Ich habe hoffentlich meinen Standpunkt klar dargelegt: Namenskonventionen, in denen versucht wird, den Typ eines Datenbank-Objekts im Namen des Objekts zu verschlüsseln, funktionieren in der Praxis nicht. Was ja noch schlimmer ist: Der Versuch, solche Konventionen einzuhalten, führt zu untragbaren Mehrkosten.

Diesen geschilderten Nachteilen stehen aus meiner Sicht keinerlei erkennbare Vorteile gegenüber. Ich verstehe nicht, warum die Administrierbarkeit einer Datenbank durch solche Konvention besser/leichter/billiger sein soll. Jeder Junior-DBA sollte in der Lage sein, sich binnen kürzester Zeit einen Überblick über die Objekte eines Typs auf einer Instanz zu verschaffen, oder aus dem Data Dictionary den Typ eines Objekts zu ermitteln!

Warum also fordern so viele Konzerne genau diese Art von Richtlinien? Geht es hier tatsächlich nur um politische Machtkämpfe zwischen unterschiedlichen Organisationseinheiten? Warum verwendet man die Zeit, die auf die Erstellung solcher Namenskonventionen verwendet (oder verschwendet?) wird, nicht dazu, sinnvolle Richtlinien, die sowohl den Entwicklern als den Betreibern eine angenehmere Zusammenarbeit ermöglichen?

Die oben gestellten Fragen sind ehrlich gemeint. Sollten Sie dazu Antworten haben, oder den von mit hier aufgezeigten Standpunkt diskutieren wollen, so bin ich für Sie gerne per E-Mail erreichbar.

Kontakt:
Michael A. Istinger
michael.istinger@gmx.at