Policy versus Legacy
Stolperfallen rund um die Virtual Private Database

Torsten Pattberg
IKB Deutsche Industriebank AG
Düsseldorf

Schlüsselworte
Datenbank, Virtual Private Database, Berechtigung, best practice, Erfahrungsbericht

Einleitung
Unsere Kredit-Plattform „KreDa“ ist eine In-House-Entwicklung bestehend aus ca. 500 Oracle Forms und 300 Oracle Reports und basiert auf beinahe 800 Tabellen. Sie wird aktuell im Rahmen des Projektes „Olymp“ mit Hilfe von Oracle ADF modernisiert. In einem bereits abgeschlossenen Teilprojekt haben wir unser System dabei mittels der „Virtual Private Database“ mandantenfähig gemacht.

Hier berichte ich über die Erfahrungen, die wir vor und nach der „Aktivierung der Policies“ in unserem Jahrzehnte lang historisch gewachsenen System gemacht haben.

Ausgangslage
Um beim Servicing von nicht-Konzern-eigenen Geschäften den strengen Compliance- und Revisions-Anforderungen zu genügen wurden wir gebeten die vorhandene Kredit-Plattform „mandantenfähig“ zu gestalten.

Als mandantenfähig […] wird Informationstechnik bezeichnet, die auf demselben Server […] mehrere […] Kunden […] bedienen kann, ohne dass diese gegenseitigen Einblick in ihre Daten, Benutzerverwaltung und Ähnliches haben. [Quelle: Wikipedia]

Technisch kamen für uns drei Alternativen in Frage:

- Aufsetzen von paralleler Hardware
- Manuelle Anpassungen der Anwendung
- Einsatz der Virtual Private Database

Nach Abwägung der Vor- und Nachteile fiel die Entscheidung auf die Virtual Private Database. Berücksichtigt wurden hierbei Aspekte wie Hardware- und Lizenz-Kosten, Entwicklungs-Aufwand, Änderungen im Deploymentprozess, Sicherheit, die Anzahl der erwarteten Mandanten und die angebundenen Schnittstellen. Nachdem der „Proof of Concept“ innerhalb weniger Tage erstellt war und wir die Mandantenfähigkeit dabei unserer Meinung nach als das „Hello World“ der VPD-Funktionalität identifiziert hatten, sind bis zur erfolgreichen Produktivsetzung dennoch zahlreiche Stolperfallen aufgetreten, die teilweise auch bei den „alten Hasen“ (vom DBA bis zum Entwickler) für Erstaunen gesorgt haben. Über die aufgetretenen Probleme und wie wir diese letztendlich lösen konnten, werde ich im Folgenden berichten.

Funktionsweise
Je nach Vorwissen im Auditorium betrachten wir mehr oder weniger ausführlich die Funktionsweise der Virtual Private Database – eingeschränkt auf unseren speziellen Anwendungsfall der „row-based-Security“. Als „best practice“ hat sich hier die Kombination mit Oracles SYS_CONTEXT

durchgesetzt, den wir als Ablageort des Mandanten für die jeweilige Datenbanksession verwenden und der durch einen Trigger beim Login entsprechend gesetzt wird. Wir sehen wie die Policies auf den Tabellen aktiviert werden und dadurch mittels der Policyfunktion jede Abfrage automatisch um die entsprechende Mandanten-Einschränkung ergänzt wird. Anschließend skizzieren wir eine einfache Lösung mit der wir dafür sorgen, dass neue Datensätze in ihrem jeweiligen Mandanten entsprechend markiert werden, ohne dass wir umfangreiche Änderungen in der Anwendung vornehmen mussten.

Datenmodell
In diesem Teil stellen wir fest, dass wir neben der obligatorischen neuen Spalte in jeder mandantenfähigen Tabelle weitere Anpassungen an unserem Datenmodell vornehmen mussten und diese leider Änderungen an unserer Anwendung nach sich zogen.

Wir starten mit den Primary-Keys, die man in zwei unterschiedliche Typen kategorisieren kann: natürliche und künstliche („surrogate“) Schlüssel. Beide kommen in unserem System vor und letztere wurden teilweise fachlich genutzt, was zu korrigieren war. Da die natürlichen Schlüssel sowie die Unique Keys (logischerweise) um den Mandanten ergänzt wurden, wirkte sich dies auf korrespondierende Foreign-Keys aus, die wir deshalb ebenfalls genauer betrachten. Probleme bereiteten hierbei insbesondere deren nicht-obligatorische Spalten. Wir stellen in diesem Zusammenhang ebenfalls fest, dass die referentielle Integrität nicht mehr ohne Weiteres gewährleistet werden kann.

Die meisten der genannten Probleme konnten wir mit Hilfe dynamisch erzeugter Skripte lösen, die in den vorhandenen Deployment-Prozess für das Datenmodell integriert wurden und die entsprechende Datenbank-Objekte (z.B. Trigger) anlegen.

Mandantenübergreifendes Arbeiten
Obwohl es den Grundsätzen der Mandantenfähigkeit widerspricht, gab es seitens der Fachabteilungen (natürlich) Anforderungen, die eine mandantenübergreifende Sicht wünschten, z.B.

- Revisionsprüfungen
- Schnittstellen (z.B. Data Warehouse)

In diesem Teil werden wir jedoch sehen, dass wir dies technisch durch eine entsprechende „Null“-Policyfunktion zwar einfach hätten realisieren können, fachlich aber eine korrekte mandantenübergreifende Arbeitsweise der Anwendung nicht sicherstellt werden kann. Gründe hierfür sind unter anderem kartesische Produkte, TOO_MANY_ROWS Exceptions und explizite Cursor, die nach den vorgenommenen Datenmodell-Änderungen ohne Berücksichtigung der Policies zu Fehlern führen würden. In diesem Zusammenhang betrachten wir auch das EXEMPT ACCESS POLICY Privilege, das die gleichen Probleme nach sich ziehen kann.

Datenbank
In diesem Teil betrachten wir Auswirkungen von Policies auf bestehende Datenbank-Objekte. Bei der Verwendung von Materialized Views stößt man auf den Fehler „ORA-30372: fine grain access policy conflict“. Dies führt unter Berücksichtigung der Resultate aus dem vorigen Teil dazu, dass die Vorzüge von Materialized Views allenfalls mit besonderer Vorsicht zu genießen sind. Gleiches gilt für einige unserer „technischen“ Datenbank-User, die aus verschiedenen Gründen ohne aktivierte Policies laufen müssen. In diesem Zusammenhang sprechen wir auch über die AUTH_ID bei der Anlage von Packages und Triggern und werden sehen, welche Auswirkungen DEFINER und INVOKER Rechte haben (bzw. nicht haben).

Performance
Während Teile der Anwendung im ursprünglichen Mandanten nach den bisher vorgenommenen Änderungen ohne Probleme liefen, verschlechterte sich die Laufzeit an manchen Stellen dramatisch. Dies galt sowohl für die Online-Anzeige von Daten als auch für Batch-Prozesse, die im Rahmen unserer Nacht- oder Ultimo-Verarbeitung laufen müssen. Neben „einfachen" Performance-Optimierungen, etwa durch „Scalar Subquery Caching" oder dem Refactoring einiger zentraler Views und Packages, war die Aktivierung des MERGE ANY VIEW Privilegs eine zielführende Maßnahme. Was es bewirkt und ob es sich hierbei um ein Sicherheits-Problem handelt, werden wir zeigen.

Vorhandene „Performance-Tricks" wie Caching-Funktionalitäten (z.B. durch assoziative Arrays), die wir an verschiedenen Stellen verwenden, prüfen wir an dieser Stelle ebenfalls auf mögliche Fehlerquellen.

ADF, Forms und Reports
User-Daten wie Name oder Team konnten aus diversen Gründen nicht mandanten-unabhängig modelliert werden. Während dies für unsere Forms-Anwendung kein Problem darstellte, musste im Rahmen der Autorisierung und Authentifizierung bei der ADF Entwicklung beachtet werden, dass bestimmte Abfragen schon vor der eigentlichen Anmeldung eines User stattfinden.

Des Weiteren betrachten wir auch die Auswirkungen, die ein Mandantenwechsel im laufenden Betrieb nach sich ziehen kann. Hier war das „stateless" ADF Vorgehen von Vorteil und Forms benötigte eine Sonderbehandlung.

Da ein Report in seiner eigenen Session läuft, mussten wir in alle unsere Reports eine entsprechende Logik bei deren Aufruf integrieren. Hier war sowohl unser bestehendes Reports-Framework als auch Oracles Reports-API hilfreich.

Daten
In diesem Teil betrachten wir einen neuen leeren Mandanten. Es gab Programme, die mit leeren Tabellen nicht zu Recht kamen und korrigiert werden mussten.

Beim Urladen kann man entweder alle Verwaltungstabellen in den neuen Mandanten „kopieren" oder wir identifizieren exakt die Teilmenge der Daten, die für die korrekte fachliche Abbildung des neuen Mandanten notwendig sind. Hier zeigt sich letztendlich wie gut IT und die Fachabteilungen die gesamte Anwendung kennen und auch wie gut diese dokumentiert ist. In beiden Fällen ist das technische Problem zu lösen, dass die künstlichen Schlüssel in verschiedenen Mandanten unterschiedlich sind und in voneinander abhängigen Tabellen entsprechend korrigiert werden müssen.
Wir werden sehen, wie wir mittels eines eigens hierfür erstellten Skript-Generators komplexe Master-Detail Beziehungen kopieren und den Inhalt von zwei Mandanten vergleichen können. Dies alles geschieht mit Hilfe eines neu aufgesetzten Repositories, in welchem wir die vorhandenen Informationen aus dem Oracle Data-Dictionary um KreDa-spezifische technische (z.B. bzgl. Mehrsprachigkeit) und fachliche (z.B. „sprechende Ids") Informationen ergänzen. Letztendlich erhalten wir hierdurch ein Paket für das einfache Aufsetzen eines neuen Mandanten, welches wir beim mittlerweile vierten produktiven Mandanten bereits erfolgreich verwenden konnten.

Schnittstellen
Im Rahmen des Projekts war natürlich sicherzustellen, dass die Zusammenarbeit mit „Drittanwendungen" wie SAP oder dem Data Warehouse funktioniert, ohne dass es etwa zu einer Vermischung von Daten verschiedener Mandanten kommt. Die entsprechenden Lösungen hängen zum

einen davon ab, ob das angesprochene System selbst mandantenfähig ist. Zum anderen ist aber auch die eingesetzte Technik der jeweiligen Schnittstelle zu berücksichtigen. Wir unterscheiden hier

- Dateien
- Datenbank-Links
- Views
- Oracle Mediator

und zeigen, wie wir diese mittels einer zentralen Konfiguration entsprechend pro Mandant steuern.

Ausblick
Die Einführung der Mandantenfähigkeit mittels der Virtual Private Database eröffnet auch aus technischer Sicht neue Möglichkeiten, insbesondere für das Thema „automatisiertes Testen". Wir zeigen im letzten Teil, wie wir mit Hilfe unseres Skript-Generators in kurzer Zeit automatisiert einen neuen leeren Testmandanten erstellen und diesen mit spezifischen Testdaten versorgen können. Diese Testdaten können dabei sogar aus Produktionsfällen erstellt werden und dabei helfen, aufgetretene Fehler schnell in einer reproduzierbaren und unabhängigen Testumgebung zu analysieren und zu beheben.

Kontaktadresse:
Torsten Pattberg
IKB Deutsche Industriebank AG
Wilhelm-Bötzkes-Straße 1
D-40474 Düsseldorf

Telefon: +49 (0) 211-8221-3254
Fax: +49 (0) 211-8221-3554
E-Mail torsten.pattberg@ikb.de
Internet: www.ikb.de