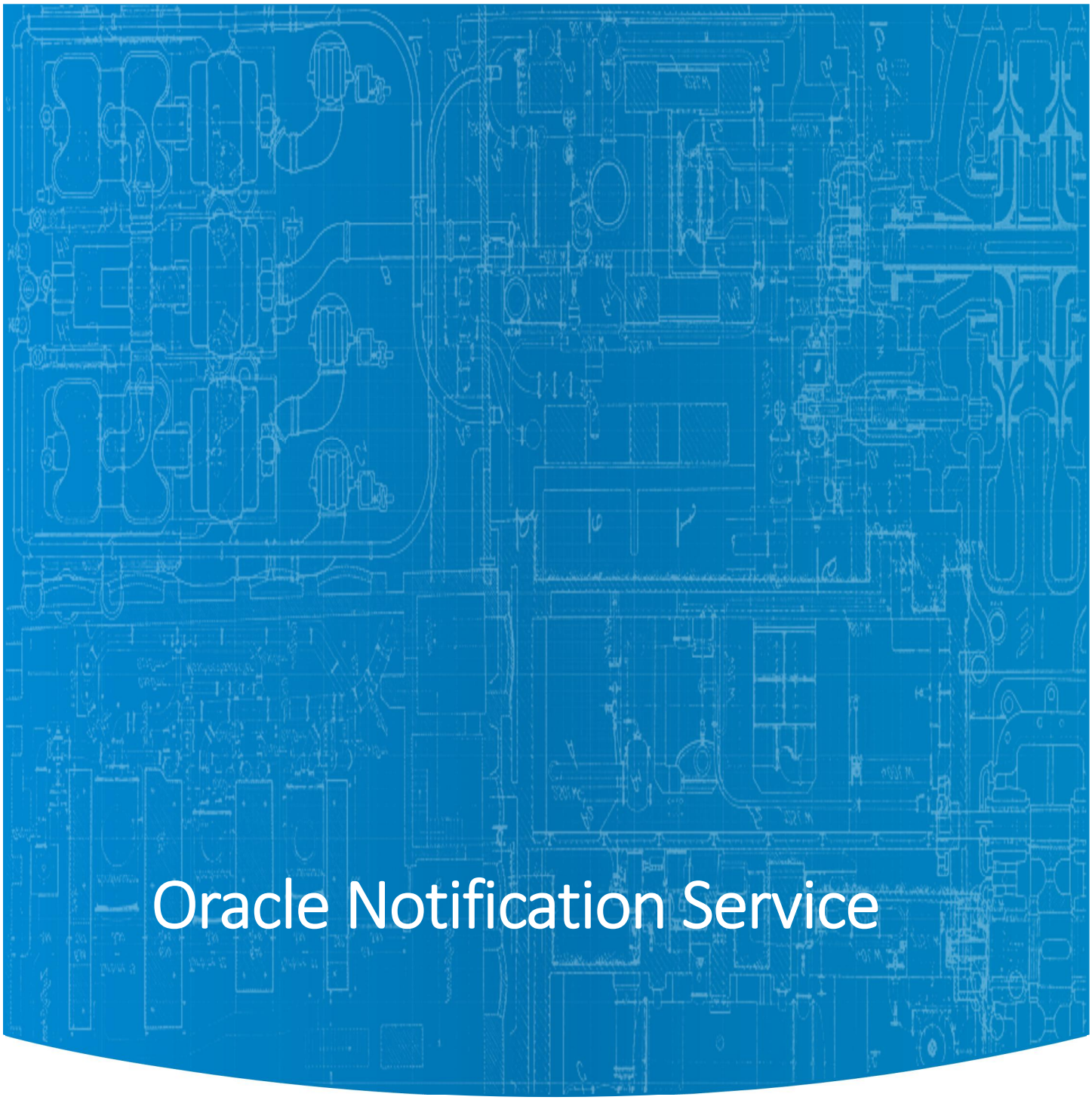

Oracle Notification Service

TECHNISCHE EVALUIERUNG

SKY4.0 GMBH – HENDRIK STILKE

SKY4.0 GMBH | Hagenring 89, 38106 Braunschweig

1 Inhaltsverzeichnis

1 Abstract / Anriss

Der *Oracle Notification Service (ONS)* ist ein Software-Paket das zum Demonstrationszweck die Notifizierungsfunktion einer Oracle Datenbank (ab Version 10g) auslotet. Wir wollen eventuelle Stärken und Schwächen einer solchen SW-Komponente evaluieren und herausfinden, ob und wie man eine solche Komponente gewinnbringend einsetzen kann. Die aktive Notifizierung von Änderungen auf einer oder mehrerer Tabellen aus einer Datenbank heraus, soll dazu genutzt werden, Nachrichten zu diesen Änderungen über einen *Enterprise Service Bus (ESB)* an Drittsysteme zu verteilen. Dieses Beispiel dient der Gegendarstellung zum derzeitigen Ansatz des Polling auf der Datenbank. Sowohl Latenzen beim Empfang von Änderungen als auch unnötige Last (wiederholtes Abfragen) sind bei einer dauerhaft laufenden Software auf einer oder gegen eine operative Unternehmensdatenbank unbedingt zu vermeiden, soweit es denn technisch möglich ist.

2 Code

Der ursprüngliche Code wurde für das Whitepaper noch etwas ausführlicher dokumentiert, als wir ihn für unseren Standard ohnehin für unabdingbar halten. Der Beispielcode zum Demonstrator finden Sie hier:

GitHub public repository: *https://github.com/Sky40GmbH/OracleNotificationService.git*

3 Architektur

3.1 Idee

Die Idee hinter diesem Demonstrator ist es, die Notifizierung der Oracle-Datenbank in einer (Java)-Komponente als Bibliothek direkt nutzbar zu machen und als einen Webservice (SOAP oder REST) anzubieten. Im besten Fall kann der Service direkt in den ESB eingebunden werden. Der OJDBC-Treiber von Oracle unterstützt Notifizierung ab Version ojdbc6. Genutzt wird hier die Version 11.2.0.4 der `ojdbc.jar`.

3.2 Umsetzung Allgemein

Da eine Service-Komponente entstehen soll, die in einer gemanagten Java-Umgebung läuft – Web-Server, Application-Server, oder ähnliches - wurde bei der Umsetzung *SpringBoot* als Framework für die Umgebung gewählt. Hiermit können sehr einfach durch Deklaration REST-Services definiert werden, die bei Start der Komponente entweder über die gemanagte Umgebung oder über einen in SpringBoot enthaltenen Tomcat Web-Server gestartet werden. Letzterer startet automatisch mit der Anwendung, falls keine Umgebung den Service hostet. Siehe hierzu die Batch-Dateien im SW-Projekt in der Ablage.

Der Demonstrator besteht aus zwei SW-Komponenten. Einem Service für die eigentliche Notifizierung und einem Client, der zeigt wie man diesen Service konsumiert. Der Client kann später durch den ESB ersetzt werden oder falls es möglich ist (d.h. falls Java-Umgebung im ESB) kann der Service auch direkt in den ESB integriert oder als Komponente (Archivformat EAR, WAR, JAR, …) dauerhaft deployed werden. Im letzten Fall kann man sich den Client sparen und die Service-Komponente umbauen.

Die Konfiguration wurde zu Vereinfachungszwecken externalisiert und wird beim Start der jeweiligen Komponente aus der Datei `application.properties` geladen.

3.3 Datenbank

Die Komponente zur Notifizierung setzt auf einer (lokalen) Oracle 11g Datenbank auf, der wir ein Schema mit einer Tabelle `aircraft` spendiert haben, in die 400.000 Datensätze eingefügt wurden (um Lasttests zu ermöglichen). Es geht hier nur um das Prinzip der Benachrichtigung. Der Zugriff ist über die Datei `application.properties` konfigurierbar. Die Datenbank ist selber aufzusetzen. Bitte darauf achten, dass der Nutzer der DB mit dem der Service sich anmeldet die Zugriffsrechte zur Notifizierung hat:

```
GRANT EXECUTE ON DBMS_CQ_NOTIFICATION TO ADMIN;

GRANT CHANGE NOTIFICATION TO ADMIN;
```

4 Übersicht Service-Komponente schematisch

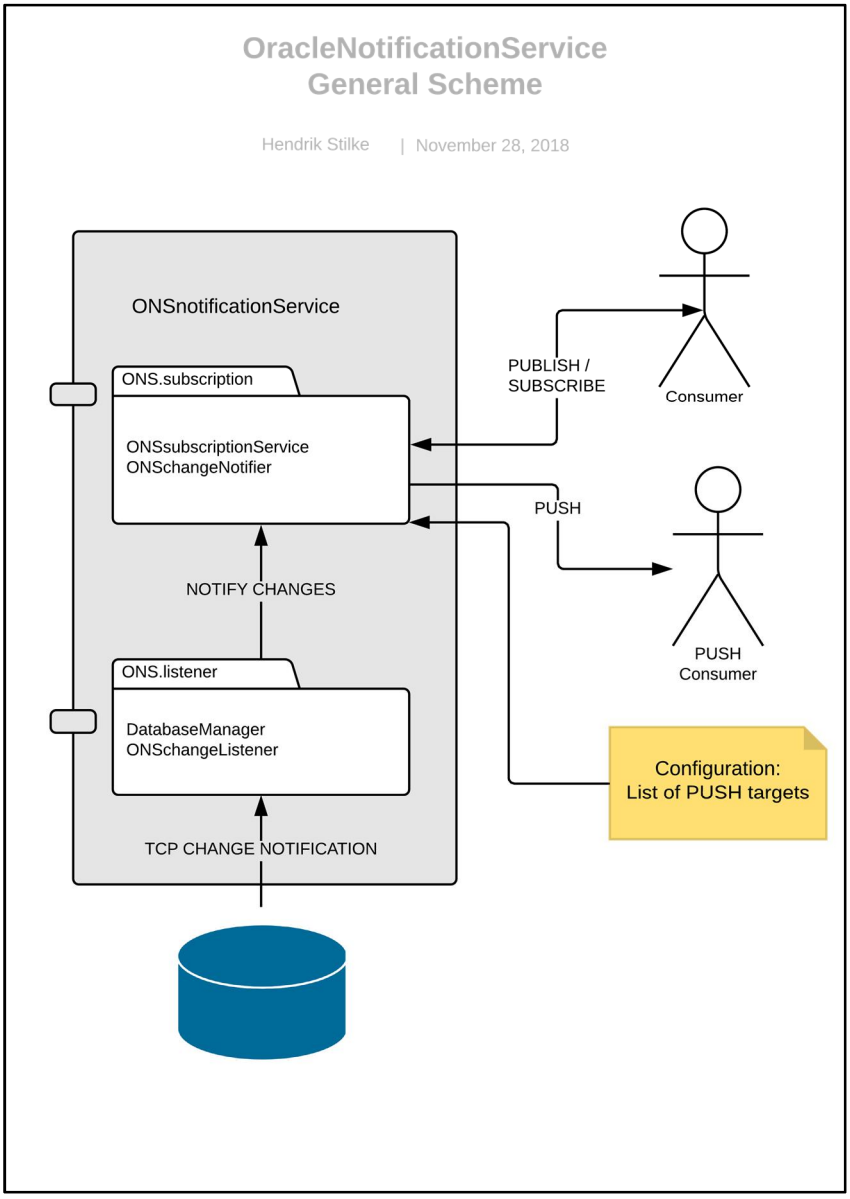

Abbildung 1: Aufbau der Komponente ONSnotificationService

Abbildung 1: Aufbau der Komponente ONSnotificationService zeigt im Diagramm das Komponentenschema der Service-Komponente. Diese besteht aus zwei Teilen: Einem Teil im Package ONS.listener der die Datenbank-Kommunikation kapselt, und einem Package ONS.subscription das die Kommunikation mit den Konsumenten übernimmt. Es können zwei Arten von Konsumenten benutzt werden. Entweder wird über Publish/Subscribe und einem Zugriffstoken die Notifizierung konsumiert oder der Konsument ist direkt in einer Liste (siehe `application.properties`) eingetragen und wird bereits ab Start des Services dauerhaft mit Nachrichten beschickt. Letztere Möglichkeit bietet sich an, damit beim Hochfahren der Systeme nach einem Ausfall der Hardware sofort wieder die Verbindung zu statischen Konsumenten in der Service-Umgebung hergestellt wird.

5 Service-Komponenten und Schnittstellen

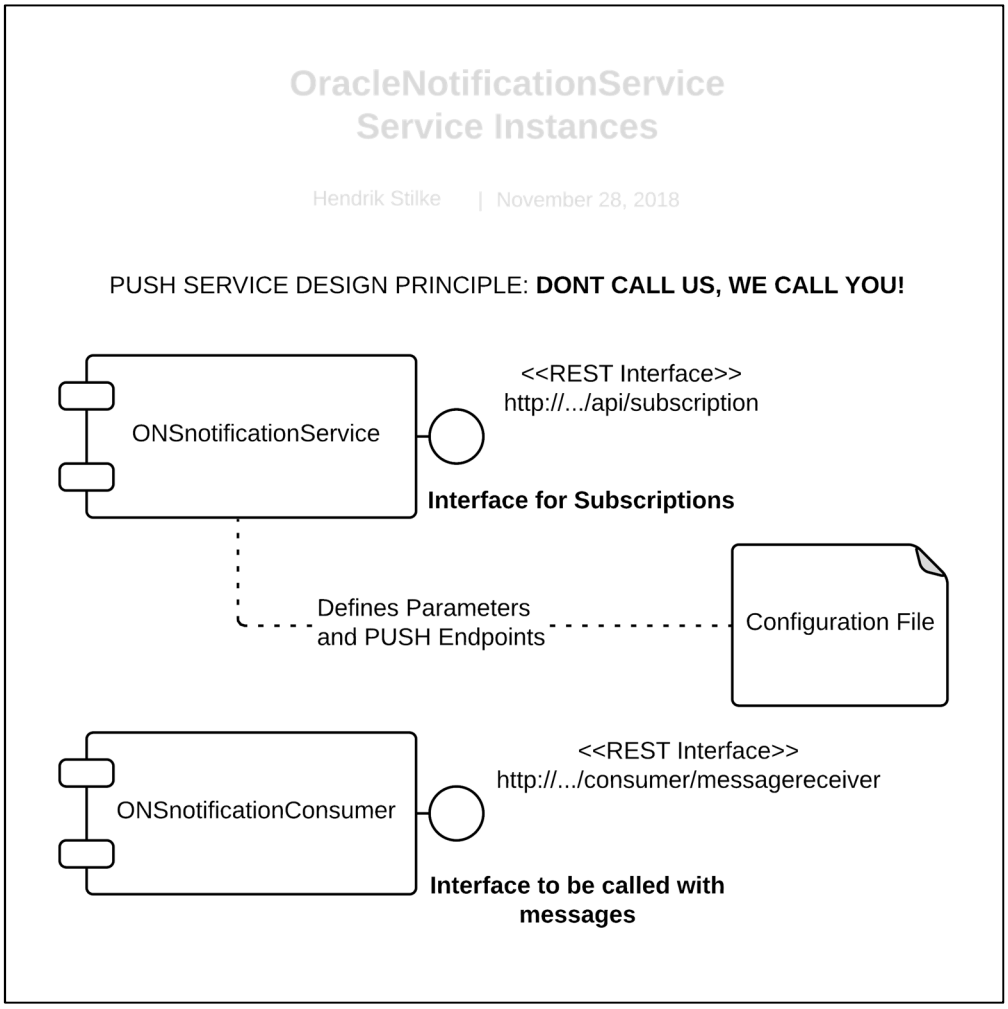

Abbildung 2: Architektur und Schnittstellen

Der Subscriber-Service stellt eine Schnittstelle für den Publish/Subscribe-Mechanismus bereit. Der Konsument ruft diese Schnittstelle auf, und übergibt dem Subscriber seinerseits einen Endpunkt bekannt auf den die Messages zum Konsumenten übermittelt werden sollen. Beides sollten REST-Interfaces sein. Design Prinzip ist die Lose Kopplung: Dont call us, we call you! (also aktives Pushen von Nachrichten).

6 Wichtige Abläufe im Detail

6.1 Start des ONS

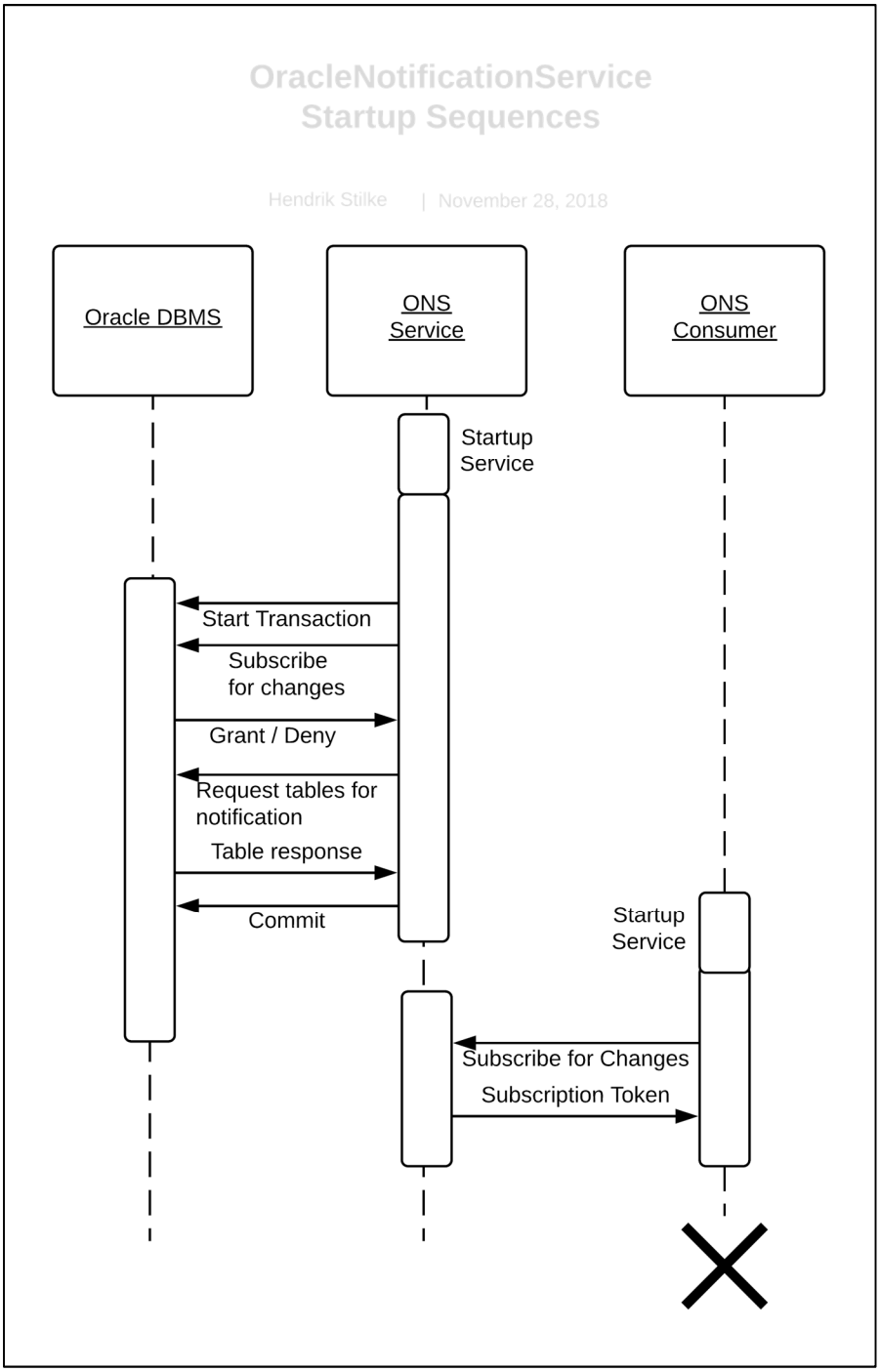

Abbildung 3: Ablauf der Service Startsequenz

Der Anmeldeprozess muss in einer Transaktion stattfinden. In der Start-Sequenz (siehe zum Start der Service-Komponente die Aufruf Batch-Datei) wird

1. die Verbindung zur Datenbank aufgebaut,
2. die Transaktion gestartet,
3. dann wird eine Benachrichtigung angefordert. Die Datenbank antwortet mit einer ID für das Abonnement (wie bei einem Session-Token).
4. Anschließend werden die Tabellen für die Notifizierung benannt, indem einmalig für jede zu beobachtende Tabelle ein `SELECT` Statement ausgeführt wird.
5. Anschließend wird die Transaktion mit `COMMIT` geschlossen.

Nicht in dieser Abbildung aber ebenfalls wichtig:

Wenn ein Abonnement nicht ordnungsgemäß bei der DB abgemeldet wurde, wird die DB bei einer erneuten Anmeldung (mit dem gleichen Port) Änderungen mehrfach abschicken. Man erkannt das daran, dass in den Nachrichten dann bei der Notifizierung mit falscher/alter Subscription-ID seitens Oracle geschickt werden. Sollte dieser Fall eintreten, werden die alten Verbindungen vom ONS Service automatisch bei der DB abgemeldet.

→ ! Wiederholtes Starten der Komponente (z.B. Tests oder Neustart App-Server) hängt sonst die alten Verbindungen und es wird mehrfach von Oracle notifiziert.

6.2 Notifizierung

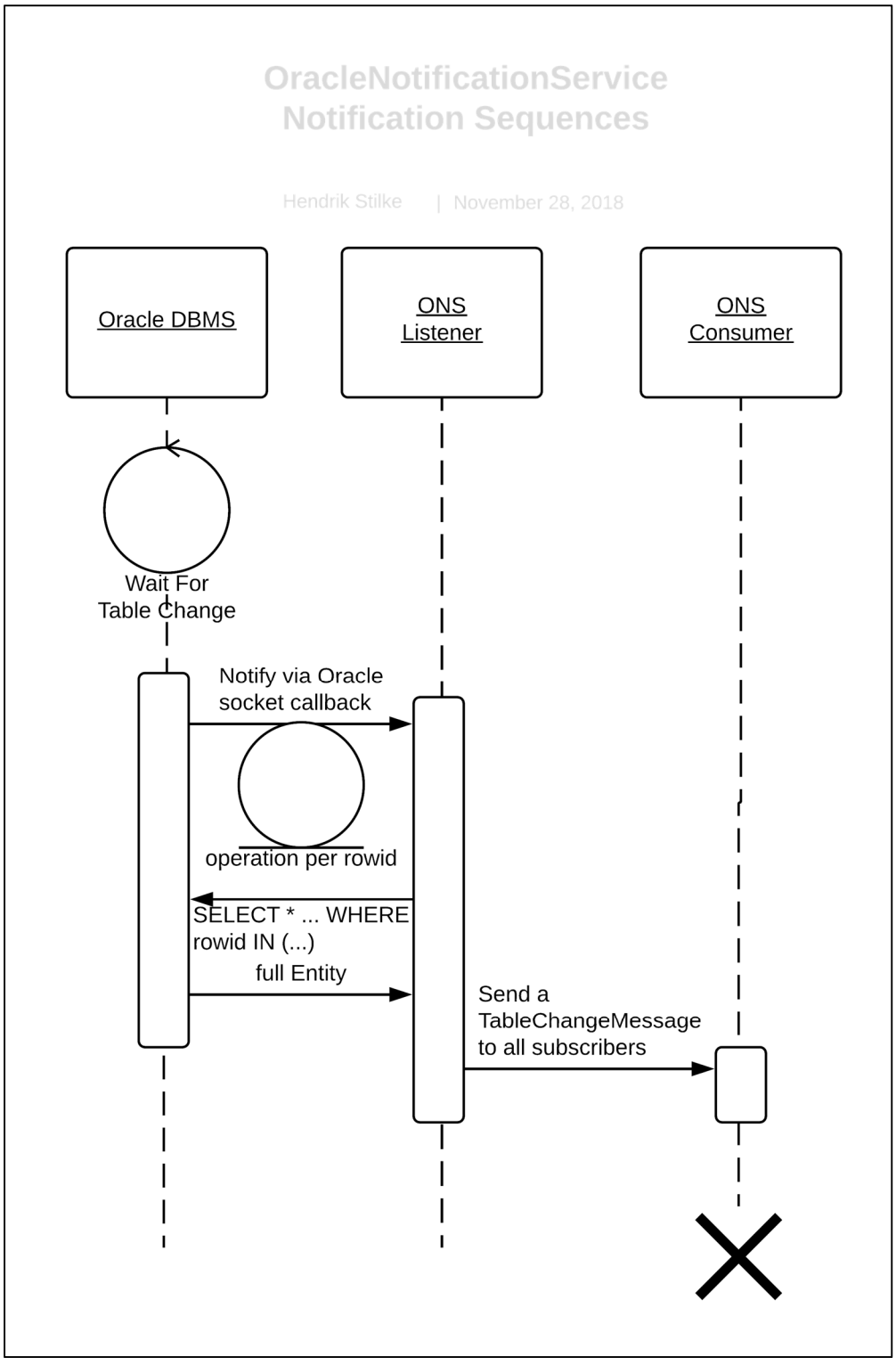

Abbildung 4: Ablauf der Notifizierung

Die Notifizierung läuft von der Datenbank aktiv (!) über eine Socket Verbindung (ojdbc6 macht das für uns) nach der Anmeldung (siehe letztes Schaubild mit dem Startup) an den ONSlistener. Achtung! Es werden nur die ROWIDs der geänderten Zeilen von der DB übertragen. Der Service muss die vollen Zeilen über die ROWIDs dann abfragen. Für gelöschte Zeilen (DELETE Kommando) kann die gelöschte Zeile nicht mehr ermittelt werden, da sie gelöscht wurde, d.h. es wird immer erst von der DB notifiziert, wenn die Transaktion der Datenbank mit COMMIT abgeschlossen wurde. Falls Löschungen als Meldungen des letzten Zustands des Datensatzes vor seiner Löschung notifiziert werden sollen, muss ggf. mit DELETE-Triggern und Schattentabellen gearbeitet werden. (Das ist nur ein Demonstrator. Wir sind uns des Problems durchaus bewusst, adressieren es hier aber nicht weiter, da die Lösung klar auf der Hand liegt.)