DBaaS mit APEX 5.1 Frontend
und Oracle Multitenant Option

Hans Eichenberger
Die Mobiliar
Bern, Schweiz

Schlüsselworte
Multitenant, APEX, RDBMS, DBaaS

Einleitung

Agile Development, DevOps, Docker, etc. - alles Schlagworte, die auch in der Mobiliar IT vor einiger Zeit Einzug gehalten haben. Mit Oracle als strategischem RDBMS-Produkt nicht ganz einfach mitzuhalten … oder doch?
Im Laufe des letzten Jahres wurde eine REST-Schnittstelle gebaut. Dank dieser kann mit dem Deployment der Applikation auch die Datenbank vollautomatisch bereitstellen.
Als zweiter Schritt sollte ein Self-Service zur Verfügung gestellt werden für PDBs gecloned von beliebigen bestehenden PDBs. Im Rahmen seiner Praxisarbeit zum Abschluss der vierjährigen Lehre hat ein IT-Azubi das APEX-Frontend in diesem Frühling realisiert.
Die noch fehlenden Teile und die Anpassung an Oracle 12.2 entstanden danach meist in Randzeiten oder während Nacht- oder Wochenend-Schichten.
Nachdem ich auf diese Themengebiete detaillierter eingegangen bin, werde ich auch noch die Frage beantworten „Wie sieht die Lösung aus und hat sich die Realisation gelohnt?“.

Automatisierte PDB-Bereitstellung via REST-Schnittstelle

Das agile Entwickeln soll letztlich nicht am manuellen, sprich langsamen, konventionellen Deployment scheitern. Darum haben wir in der Mobiliar vor ca. eineinhalb Jahren eine Lösung gesucht, wie Datenbanken möglichst automatisiert und schnell bereitgestellt werden können.
Relativ schnell stellte sich heraus, dass die Multitenant-Option, die seit Oracle 12.1 verfügbar ist, die Anforderungen an die DBs im agilen Umfeld am besten abdecken kann. Mit der Modularisierung der Applikation, sprich dem Schneiden der bestehenden „Monolithen“ in mehrere Microservices, wurde seitens der Architektur definiert, dass zwischen Microservice zu DB eine 1:1-Beziehung bestehen soll. Das bedeutet auf der einen Seite eine Entflechtung der Schnittstellen auf DB-Ebene, auf der andern Seite steigt aber die Zahl der notwendigen DBs stark an. Mit der Multitenant Option kann man diese Anforderung abdecken, ohne dass man die Server-Ressourcen aufstocken muss.
Die Randbedingungen, wie eine solche DB auszusehen hat, waren also klar. Jetzt ging es noch darum, diese DB in quasi no-time zur Verfügung zu stellen. Als Lösungsansatz haben wir eine Container-DB (CDB) gebaut und darin eine Template-PDB erstellt. Diese enthielt bereits alle standardmässigen Komponenten (App-Tablespace, App-User-Rolle, etc.). Via Rest-Schnittstelle wird auf eine zentrale Verwaltungs-DB connected und eine Funktion aufgerufen, die im Hintergrund via DB-Link auf die entsprechende CDB connected. Dort ruft sie via Scheduled-Job die Host-Scripts: clone_pdb_4_app, add_oid_entry, register_pdb_in_oem auf. Die aufrufende Schnittstelle erhält zum Schluss das Resultat, ob die PDB fehlerfrei angelegt werden konnte oder – im Fehlerfall – Informationen zum aufgetretenen Fehler.
So haben wir die automatische PDB-Erstellung in den Build/Deployment-Prozess integriert:

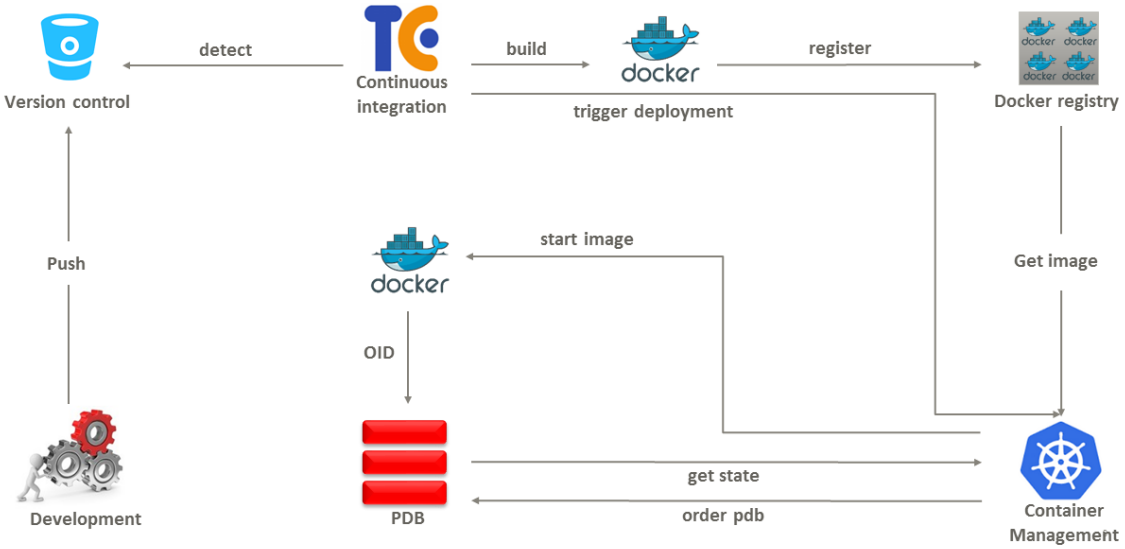
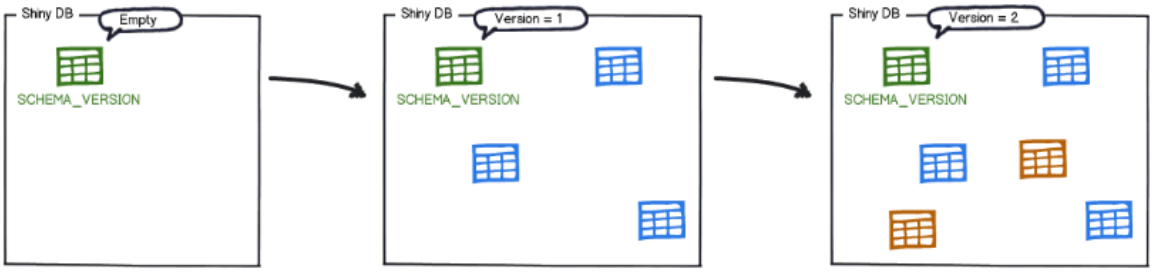


Abbildung 1: Schematische Darstellung der DBaaS-Architektur in der Mobiliar

Automatisches DB-Deployment der Applikation
Parallel zur automatischen Erstellung von PDBs haben wir auch ein Tool evaluiert, damit die Applikations-Deployments sicher und vollständig und wenn möglich automatisiert auf die Datenbanken appliziert werden können. Für diesen Teilaspekt haben wir u.a. das Open-Source-Tool flyway näher unter die Lupe genommen. Die Mobiliar setzt dieses Tool mittlerweile für die DB-Deployments der Microservices, aber auch für konventionelle Applikationen vollautomatisch ein. Mit dem Applikations-Deployment verteilen wir auch den zugehörige Scriptsatz mit den DB-Deployments. Beim Applikationsstart prüft flyway als erstes, ob die vorhandenen Scripts bereits auf der DB sind oder ob diese noch zum Teil laufen müssen. In letzterem Fall werden die fehlenden Scripts in der richtigen Reihenfolge auf der DB appliziert.



Interaktive PDB-Bestellung via APEX-Frontend
Die automatische Erstellung nach definierten Vorgaben funktionierte, der PoC wurde nahtlos in Produktion überführt. Da stellte sich schnell die Frage: Warum stellen wir nicht die Möglichkeit zur Verfügung, PDBs nach Bedarf erstellen bzw. clonen zu können, z.B. für spezielle Tests mit einem bestimmten Datenstand?
Eigentlich sollte unser IT-Lernender zum Lehrabschluss eine andere Praxisarbeit machen. Aus dieser wurde jedoch nichts – und so wurde die Idee zur Erweiterung des DBaaS-ansatzes so weit konkretisiert, dass daraus die neue Abschlussarbeit resultierte: Erstellung eines APEX-Frontends. Bereits bei der initialen Aufgabenstellung wurde klar, dass einige Vorgaben noch nicht mit 12.1 realisiert werden können; die endgültige Umsetzung musste bis 12.2 warten. Der wichtigste Grund für diese Verspätung war, dass mit 12.1 die PDBs während des Clonens read-only sind - undenkbar wenn eine aktiv genutzte PDB gecloned werden soll!

Die APEX-Architektur in der Mobiliar

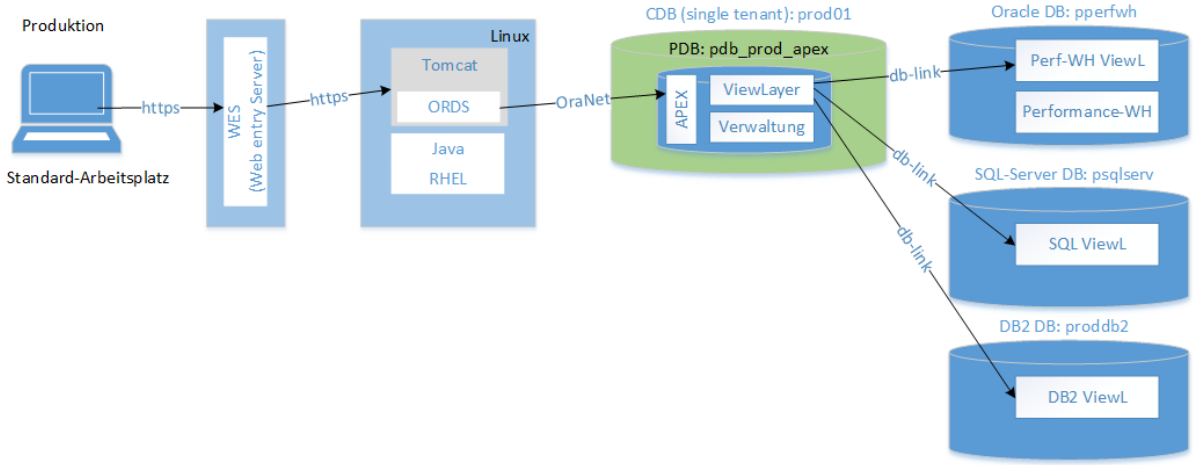


Abbildung 2: APEX-Architektur in der Mobiliar

Alle Zugriffe ab dem Mitarbeiter-Notebook erfolgen über den WES (Web Entry Server). Diese Verbindung wird mit individuellen Zertifikat des Mitarbeiters geprüft. Dadurch weiss der WES jederzeit, welcher User die Verbindung aufbaut. Dieser Umstand wird genutzt für die Single-Sign-On-Lösung der APEX-Applikationen. Der WES ergänzt den http-header mit einer definierten Variablen mit dem Usernamen als Wert. In APEX wird dieser Wert in eine APEX- intern verwendbare Variable übernommen. Eine Anmeldung mit Username/Password bzw. eine Autorisierung via LDAP ist darum nicht mehr notwendig.

Der APEX-Listener bzw. ORDS wird mit Tomcat betrieben, der Datenbankteil ist in einer PDB installiert. Ab dieser DB werden via DB-Link die Daten von Oracle-, SQL-Server- und DB2-DBs angesprochen und mit APEX visualisiert. Die zentrale PDB nutzen wir u.a. auch für die Verwaltung der automatischen PDB-Bestellung. Sowohl für die automatische wie auch die neue interaktive PDB-Bestellung greifen wir ab dieser DB auf die jeweiligen Quell- bzw. Ziel-CDBs zu.

Die Grafik zeigt schematisch die an der PDB-Bestellung beteiligten Komponenten:

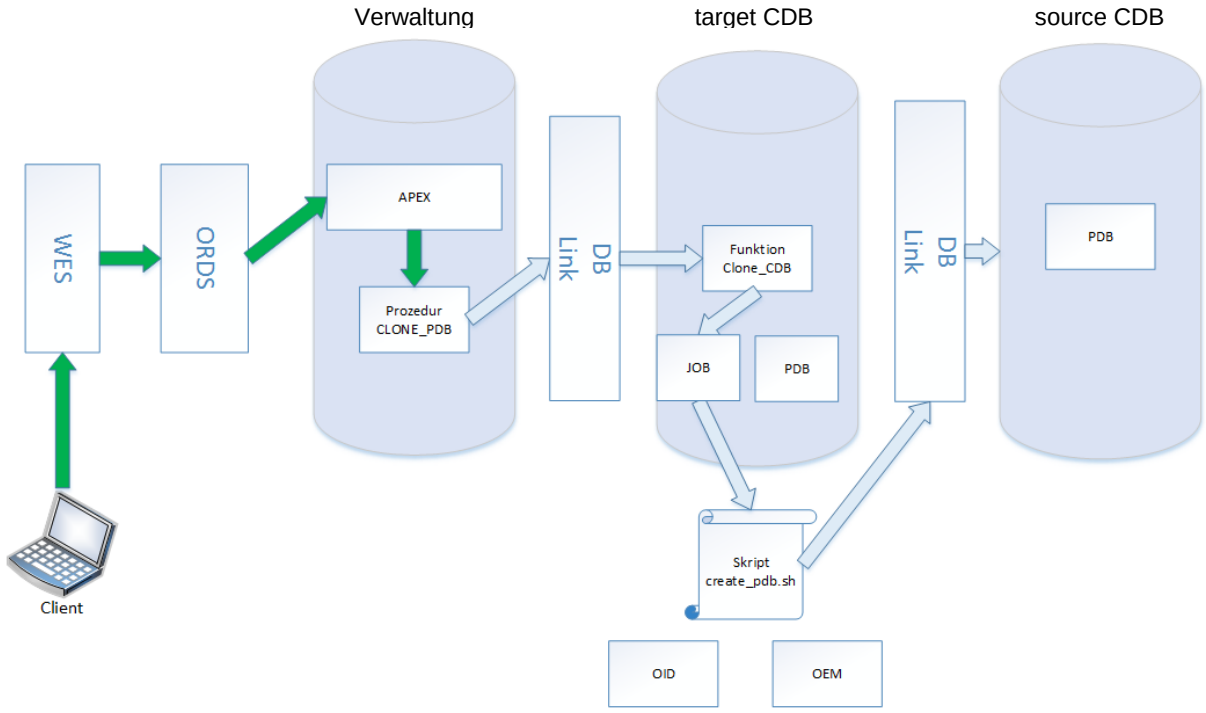


Abbildung 3: An PDB-Bestellung beteiligte Komponenten

Ablauf PDB-Bereitstellung:

Wenn alle Voraussetzungen erfüllt sind, wird in der APEX-DB eine Prozedur aufgerufen, die via DB-Link eine weitere Prozedur in der Ziel-CDB aufruft. Diese Prozedur wiederum startet einen Job, der mit den entsprechenden Parametern ein Shell-Script auf OS-Ebene startet. Dieses Script ist schlussendlich dafür besorgt, dass

- ein DB-Link auf die CDB, die die zu clonende PDB enthält, erstellt wird,
- das create pluggable database from source_pdb@dblink ausgeführt wird,
- Modifikationen an der geclonten PDB gemacht werden (z.B. Passwort-Wechsel)
- und zum Schluss der connect-string in OID ergänzt wird und dafür gesorgt wird, dass die PDB im Oracle-Enterprise Manager ersichtlich ist.

Schwachstellen der Multitenant Option mit 12.1 / Verbesserungen in 12.2

Wie schon erwähnt fehlt in 12.1 die Möglichkeit, bestehende PDBs online zu clonen.
Solange keine offenen Transaktionen in der Source-PDB vorhanden sind, startet der Clone, die PDB ist aber während des Clonens blockiert, sprich sie ist read-only.
In der Mobiliar setzen wir dataguard für den Schutz der produktiven Datenbanken ein.
Mit 12.1 hat das create pluggable database … die Standby-Database schlicht unbrauchbar gemacht. Nur mit sehr umständlichen manuellen Eingriffen wäre es überhaupt möglich gewesen, einen Neu-Aufbau der Standby-Database zu verhindern. Mit 12.2 gibt's auch in diesem Bereich deutliche Verbesserungen, die transparente Nachführung der Standby-Database ist aber noch nicht vollständig vorhanden.

Exkurs: Die Ausbildung zum Informatiker

Kurz zur Informatiker Lehre in der Schweiz: Im Rahmen des dualen Bildungssystems gibt's die Möglichkeit, nach der obligatorischen Schulzeit eine 4-jährige Lehre als Informatiker zu absolvieren. Nach einer gemeinsamen Einführung für alle Lernenden wird die Ausbildung je nach gewählter Vertiefungsrichtung abgeschlossen. Als Teil der Lehrabschlussprüfung muss während 10 Arbeitstagen eine Aufgabenstellung mit möglichst allen Aspekten eines Informatikprojektes bearbeitet werden. Diese Arbeit wird als Integrierte praktische Arbeit bezeichnet, oder kurz IPA.

Einbindung der Berufslehre im schweizerischen Bildungssystem:

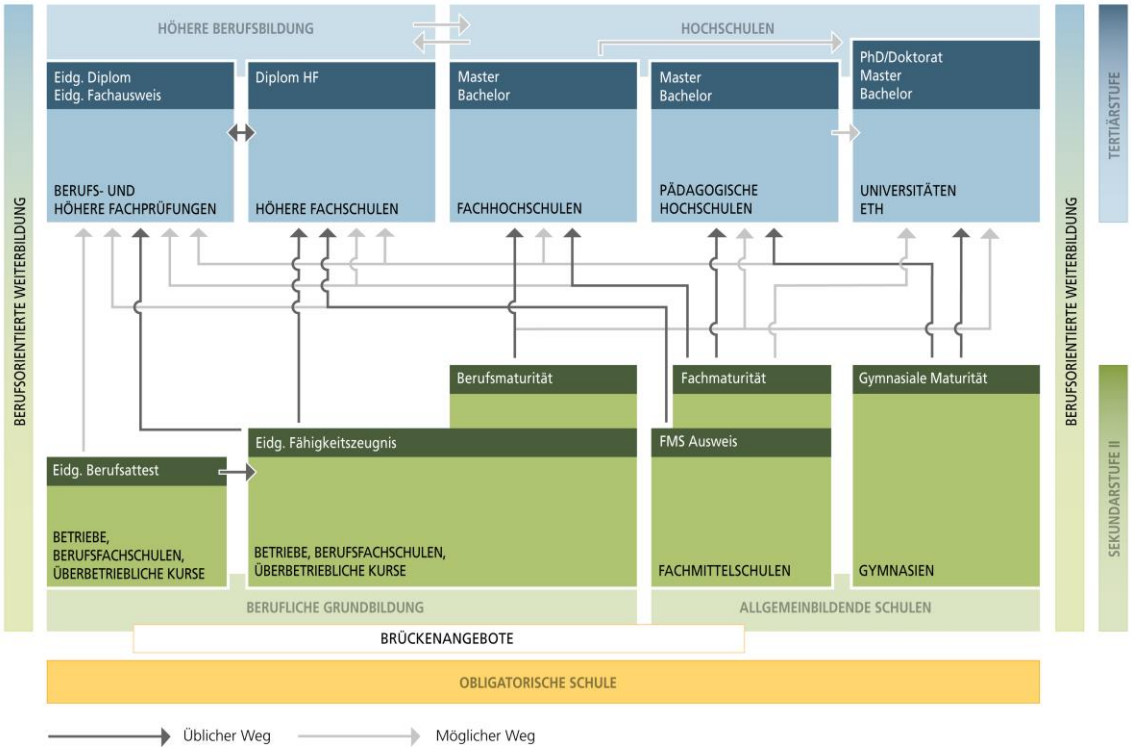


Abbildung 4: Bildungssystem der Schweiz

Ergebnis der IPA

In zehn intensiven Tagen Entwicklungs- und Projektarbeit ist ein APEX-Frontend entstanden.
Dieses erlaubt es, je nach Funktion des Anwenders PDBs als Clone aus einer fix vorgegebenen Auswahl an Template-PDBs zu bestellen. Als DBAs haben wir die Möglichkeit, PDB-Bestellungen mit speziellen Wünschen weiter zu bearbeiten oder abzulehnen. Die Backend-Funktionalität war von Anfang an erst im Nachgang zur IPA eingeplant. Die Zielvorgaben der IPA konnten grösstenteils erfüllt werden: Die von Experten geprüfte IPA wurde mit Erfolg abgeschlossen!

Erkenntnisse zum Projekt

Wie dies für Vorhaben innerhalb des IT Betriebs üblich ist, wurde ein grosser Teil umgesetzt, ohne als eigentliches Projekt im Sinne der Project Governance unseres Unternehmens deklariert gewesen zu sein. Dies führte dazu, dass die Prioritäten häufig auf andere Projekte gelegt werden mussten. Das hatte u.a. zur Folge, dass die Realisierung länger dauerte als ursprünglich angedacht.
Sicher sehr positiv war die Durchführung eines Teilprojektes als IPA: Alle Beteiligten konnten Ihre Anforderungen so klar definieren, dass innerhalb von 10 Tagen das Frontend für die PDB-Bestellung realisiert werden konnte. Gerade das Dokumentieren der Anforderungen war sehr hilfreich: Damit werden allfällige Schwachpunkte schon früh erkannt – man kann sie im Idealfall eliminieren.

Hat sich der Aufwand für dieses Projekt gelohnt?

Wer die Multitenant Option für Microservices oder ähnliches einsetzt, braucht sehr schnell einen Automatismus zum Erstellen von PDBs. Für diesen Zweck ist eine interaktive PDB-Bestell-App sicher nicht das Richtige.

Ob sich die Entwicklung einer eigenen interaktiven PDB-Bestellung lohnt, müssen Sie sicher im Einzelfall für Ihr Unternehmen prüfen. Was ich sicher empfehlen kann: stellen Sie die Anforderungen zusammen. Aus diesen lassen sich Standards ableiten und Vorbereitungen treffen, wie z.B. die Bereitstellung von sinnvollen Template-PDBs oder auch die Abbildung von Prozessen mit Scripts. Diese Scripts kann man bei Bedarf manuell anstossen, danach laufen jedoch alle notwendigen Schritte ohne weitere manuelle Intervention ab, bis die PDB zur Nutzung bereitsteht.

Ein sehr wichtiger Punkt mit Mulititenant ist es, den Überblick zu behalten und dazu eine zentrale Verwaltung der PDBs zu unterhalten mit mit Zusatzinformationen wie z.B. „Welche Applikation nutzt welche PDB?“, „Wer ist für die Applikation zuständig?“ etc., am besten mit einer Möglichkeit, die Daten zu visualisieren – das halte ich für essentiell.

Fazit: Für die Mobiliar hat sich die Entwicklung dieser App gelohnt, vor allem die erwähnten Zusatznutzen sind für uns sehr wertvoll.

Kontaktadresse:
Hans Eichenberger
Schweizerische Mobiliar Versicherungsgesellschaft AG
Monbijoustrasse 68
CH-3001 Bern

Telefon:	+41 (0) 31-389 69 88
E-Mail	hans.eichenberger@mobiliar.ch
Internet:	www.mobiliar.ch