Mit einem Katzensprung von Oracle zu PostgreSQL?

Wie die Migration in die AWS-Cloud besser gelingt

Christian Ballweg, Opitz Consulting Deutschland

„Das Unerwartete zu erwarten, zeigt einen durch und durch modernen Geist." (Oscar Wilde). Die Migration von Oracle auf Open-Source-Alternativen – besonders PostgreSQL – ist derzeit in aller Munde. Dieser Artikel gibt eine Übersicht über eine gute Vorbereitung, den Migrationsweg und mögliche Herausforderungen.

Eine Migration von Oracle zu PostgreSQL wird besonders von den Anbietern der Tools und Zielumgebungen oft als recht einfach dargestellt. Das dient der Übersicht und schafft auch Vertrauen in die Darstellung. Denn wer will hier schon mit Problemen empfangen werden? Leider stellt sich oft erst im Verlauf einer Migration zu dem Open-Source-Datenbanksystem heraus, was man noch hätte untersuchen können, was vergessen oder falsch verstanden wurde – oder dass eine Migration durch falsch oder nicht kalkulierte Aufwände und Risiken am Ende komplexer und damit teurer wird als ursprünglich angenommen. Deshalb sollten der Migrationsweg gut vorbereitet sein und mögliche Herausforderungen eingeplant werden.

In diesem Artikel will ich auf einzelne Schritte und interessante Aspekte einer Migration eingehen. Dadurch erhalten Sie eine empfohlene Methode, wie Sie eine Migration von Oracle nach PostgreSQL planen und durchführen können.

Tatsächlich kann alles glatt laufen. Wir erleben bei unserer Kundschaft immer wieder zwei Extrempunkte bei der Migration:

1. Einfach und schnell: Manchmal gelingt der Übergang zu PostgreSQL automatisiert und ohne Probleme, wenn wir mithilfe des AWS Schema Conversion Tool (SCT) das Zielschema aus der Oracle-Quelle nach PostgreSQL übersetzen. Wir schauen noch einmal über die vorgeschlagenen Datentypen und schließen die Migration mit der Befüllung des Zielschemas durch den AWS Database Migration Service (DMS) ab. Anschließend aktivieren wir noch die Trigger und Foreign-Key Constraints, die eine parallele Befüllung per DMS gegebenenfalls verhindert hätten. Die Applikation ist dabei einfach gehalten und bedarf keiner weiteren Anpassung als einer minimalen Umstellung des „DB-Dialekts".
2. Unerwartet kompliziert: Die Datenbank besteht aus vielen nicht automatisch konvertierbaren Codebestandteilen und komplizierter Logik, deren Analyse und Umstellung viele Personentage, Wochen oder sogar Monate verschlingen kann – und schlimmstenfalls durch unsachgemäße Umstellung unvollständige oder falsche Ergebnisse in schlechter Performance liefert.

Ob es einfach oder kompliziert wird, können Sie im Vorfeld nicht sicher wissen. Doch das heißt nicht, dass Sie einen Blindflug riskieren sollten. Denn der kann unter Umständen teuer werden. Deshalb ist es hilfreich, die wichtigsten Herausforderungen zu kennen und Ihre Migration bewusst zu steuern.

Doch bevor wir nun direkt in Details springen und uns über mögliche Probleme und deren Lösung Gedanken machen, sollten wir einen Schritt zurücktreten und uns das Gesamtvorhaben ansehen.
Im Sinne von

„Wenn ich eine Stunde Zeit hätte, um ein Problem zu lösen, würde ich 55 Minuten damit verbringen, über das Problem nachzudenken und 5 Minuten über die Lösung."
Albert Einstein

Stellen Sie die Frage nach dem Warum

Eine Migration von einer zur anderen DB-Engine und dazu noch vom eigenen Rechenzentrum in die Cloud ist selten trivial. Zwei Fragen sollten Sie sich im Vorfeld stellen: Welche Ziele verfolgen Sie? Und: Welche Aufwände wollen Sie investieren, um diese Ziele zu erreichen?

Mit der STAR-Methode *[1]* können Sie sich den Antworten Schritt für Schritt annähern:

S **Situation:** Warum widmen Sie sich diesem Vorhaben? Welches Umfeld ist betroffen und wie sieht es aus?
T **Task:** Welche Probleme möchten Sie lösen? Welche (kurz- und langfristigen) Ziele verfolgen Sie?
A **Action:** Was müssen Sie zur Zielerreichung tun? Welche Werkzeuge können Sie dabei unterstützen?
R **Result:** Wann sind diese Ziele erfüllt? Wann würden Sie die Reißleine ziehen?

Wann ist PostgreSQL eine Alternative?

Diese drei Auslöser für die Migration zu PostgreSQL erleben wir in der Praxis am häufigsten:

- Wenn Upgrades nötig werden … stehen oft größere Veränderungen ins Haus, zum Beispiel wenn ein Produkt wie Oracle <19c nicht mehr supportet wird und viele beginnen, an einen Wechsel zu denken.
- Wenn die Lizenzkosten explodieren … überlegen viele, zu Open-Source-Produkten zu wechseln.
- Wenn Wildwuchs eingedämmt werden soll …, weil beispielsweise zu viele verschiedene Systeme unterschiedlicher Anbieter im Einsatz sind, kann die Kon-

Abbildung 1: Vorhaben unterteilen (Quelle: Christian Ballweg)

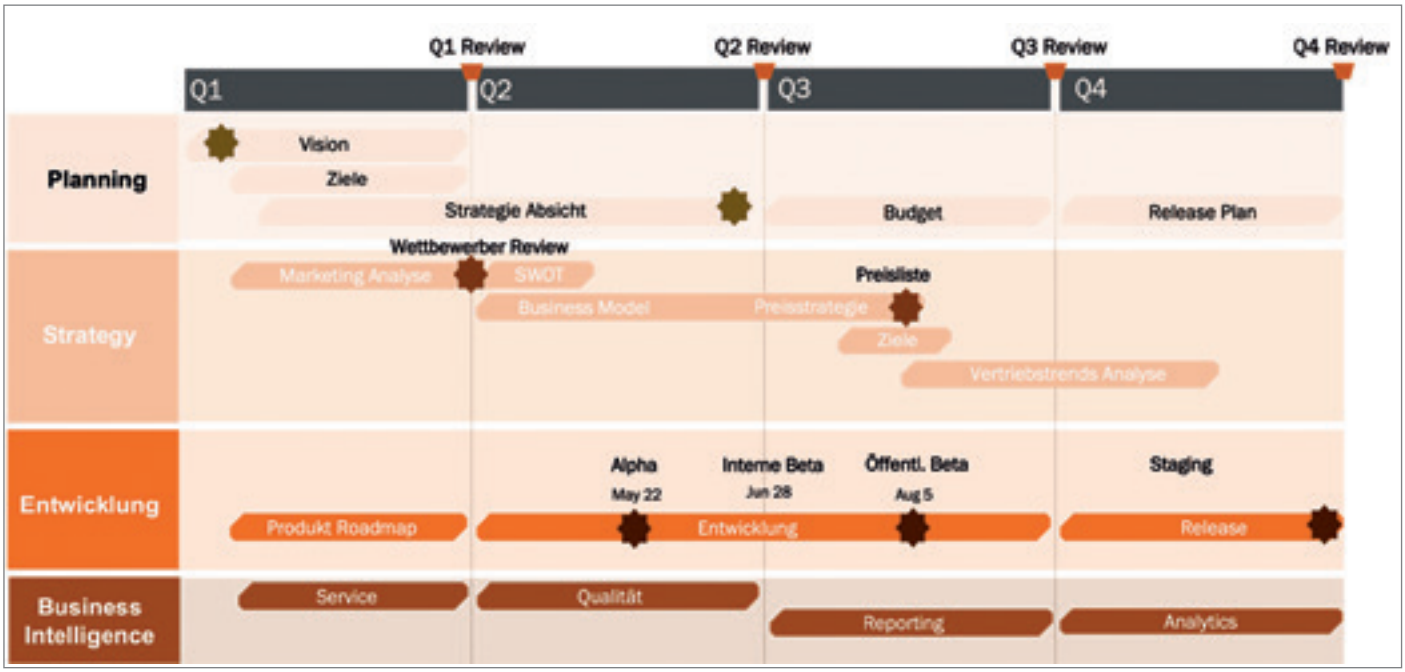

Abbildung 2: Roadmap-Beispiel aus ©projekte-leicht-gemacht.de [2]

solidierung auf ein anderes Produkt erwogen werden.

Mit einer Migration zu PostgreSQL verfolgen Sie zudem möglicherweise folgende strategischen Ziele:

- Modernisierung und Einführung einer skalierbaren Architektur
  - Virtualisierung oder Konsolidierung auf Hyper Converged Infrastructure
  - Containerisierung mit Docker/Kubernetes
  - Automatisierung und Dynamisierung (CI/CD Deployment inklusive DB)
- Datenbankmigration in die Cloud
  - als Managed Service
  - auf selbst verwaltete (virtuelle) Server

Warum PostgreSQL für die Erreichung dieser Ziele eine gute Wahl darstellen kann, ist bereits vielen DOAG-Vorträgen und Blog-Artikeln zu entnehmen. Exemplarisch erwähnen lassen sich neben Lizenzkosten-Ersparnissen auch die vielfältige Einsetzbarkeit, Erweiterbarkeit und Robustheit, bei gleichzeitig schlankem und Ressourcen-schonendem Aufbau und damit einhergehender guter Performance von PostgreSQL.

Um nicht im erwähnten Blindflug durch ein Projekt zu stolpern, ist eine gute Vorbereitung nötig. Immerhin sind nicht nur DBAs daran beteiligt, sondern auch Zuständige für Development, Architecture und so weiter bis hin zur Fachabteilung, mit Mitarbeiter:innen, die Ihre Dateninhalte kennen und nutzen. *Abbildung 1* zeigt die drei Bereiche, in die Sie ein Migrationsvorhaben unterteilen können.

1. Roadmap

Wie bei jeder Vision, die Sie in die Praxis umsetzen wollen, erstellen Sie auch bei der Migration zu PostgreSQL zunächst eine Roadmap *[2]*. Darin visualisieren Sie die einzelnen Meilensteine *(Abbildung 2* zeigt hierzu ein Beispiel), die später im Projektplan mit den notwendigen Details konkretisiert und fortgeschrieben werden.

Die Roadmap gibt einen Überblick über das gesamte Projekt, ohne zu sehr ins Detail zu gehen. Sie hilft Ihnen, Zwischenziele zu formulieren und sich einen ersten Überblick über den zeitlichen Ablauf zu verschaffen.

Vielleicht fragen Sie sich, warum es so wichtig ist, einen einfachen Überblick zu bekommen?

Hier geht es um mehr als die Abläufe der Migration selbst. Von der Datenbank, die Sie migrieren, sind viele weitere Systeme abhängig, zum Beispiel

- Anwendungen, die die Datenbank direkt nutzen
- ETL-Prozesse, die Daten in die Datenbank laden oder aus ihr extrahieren
- andere Anwendungen, die etwa über Datenbanklinks die Datenbank als Integrationsschnittstelle nutzen

Außerdem sind die Modernisierungsaktivitäten der einzelnen Stakeholder zu koordinieren. Deshalb ist die Erarbeitung einer Roadmap eine nicht zu unterschätzende Aufgabe.

Die Roadmap dient also als Grundlage für eine detaillierte Programm- und Projektplanung. Dieser mit den zuvor gesammelten Fakten auszuarbeitende Projektplan ist detaillierter. Er enthält konkret benannte Aufwände, die Ressourcenplanung sowie Termine für die Umsetzung.

Wie wir bei Opitz Consulting vorgehen, wenn wir eine Roadmap erstellen, zeigt das „OC-Vorgehensmodell für die Erstellung einer Roadmap“ in *Abbildung 3*.

Da technische und fachliche Analysen gegenseitig aufeinander einwirken, sind für eine detaillierte Erarbeitung im Rahmen der Projektplanung und späteren Durchführung mehrere Iterationen notwendig (vgl. *Abbildung 4*).

Was brauchen Sie, um herauszufinden, ob eine Migration generell durchführbar ist und welche Werkzeuge und Zielarchitektur Sie wählen sollten? Darum geht es im Pre-Assessment. Werkzeuge wie der „Migration Assessment Report“ des SCT sowie vorbereitete Aufgabenpakete und Fragen helfen Ihnen, die Antworten zu finden.

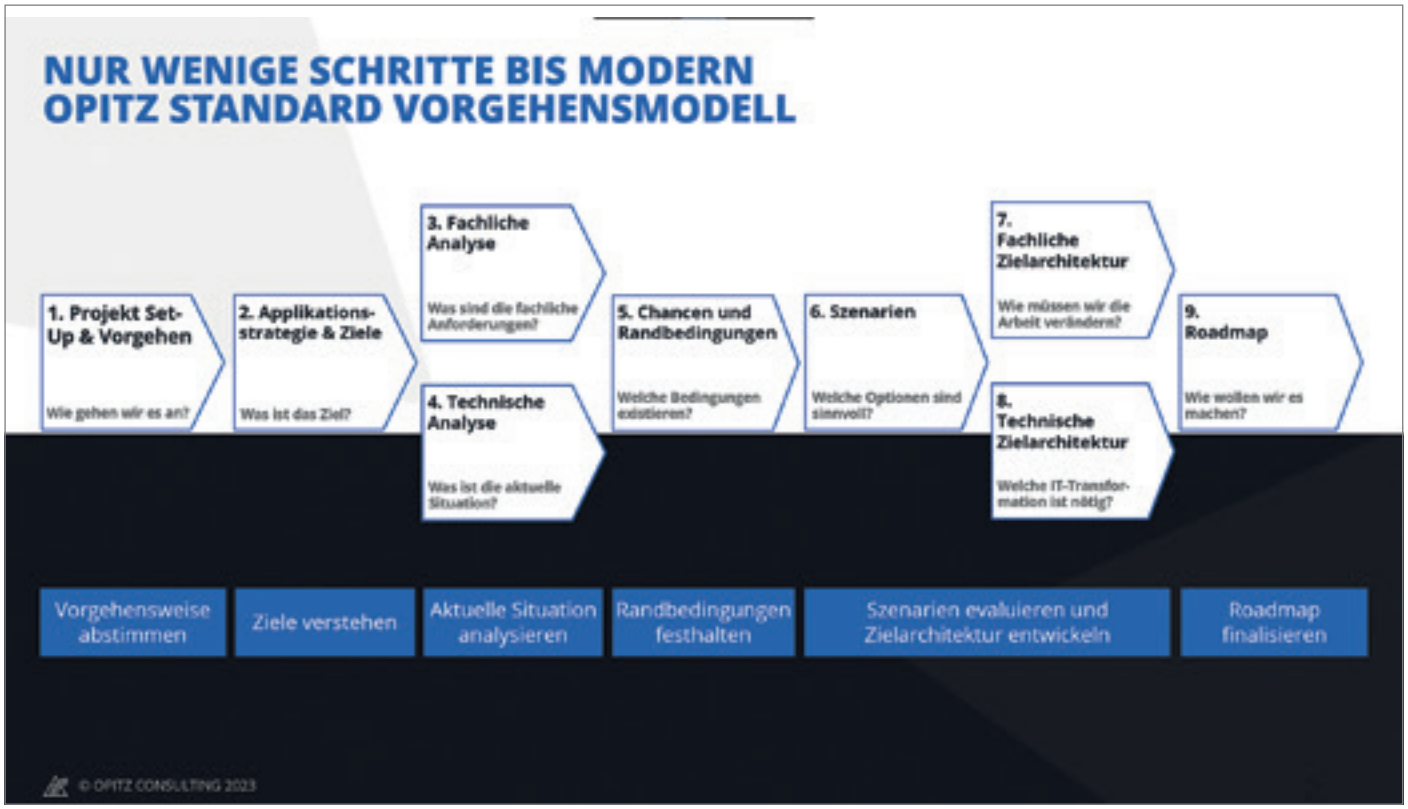

Abbildung 3: OC-Vorgehensmodell (©Opitz Consulting)

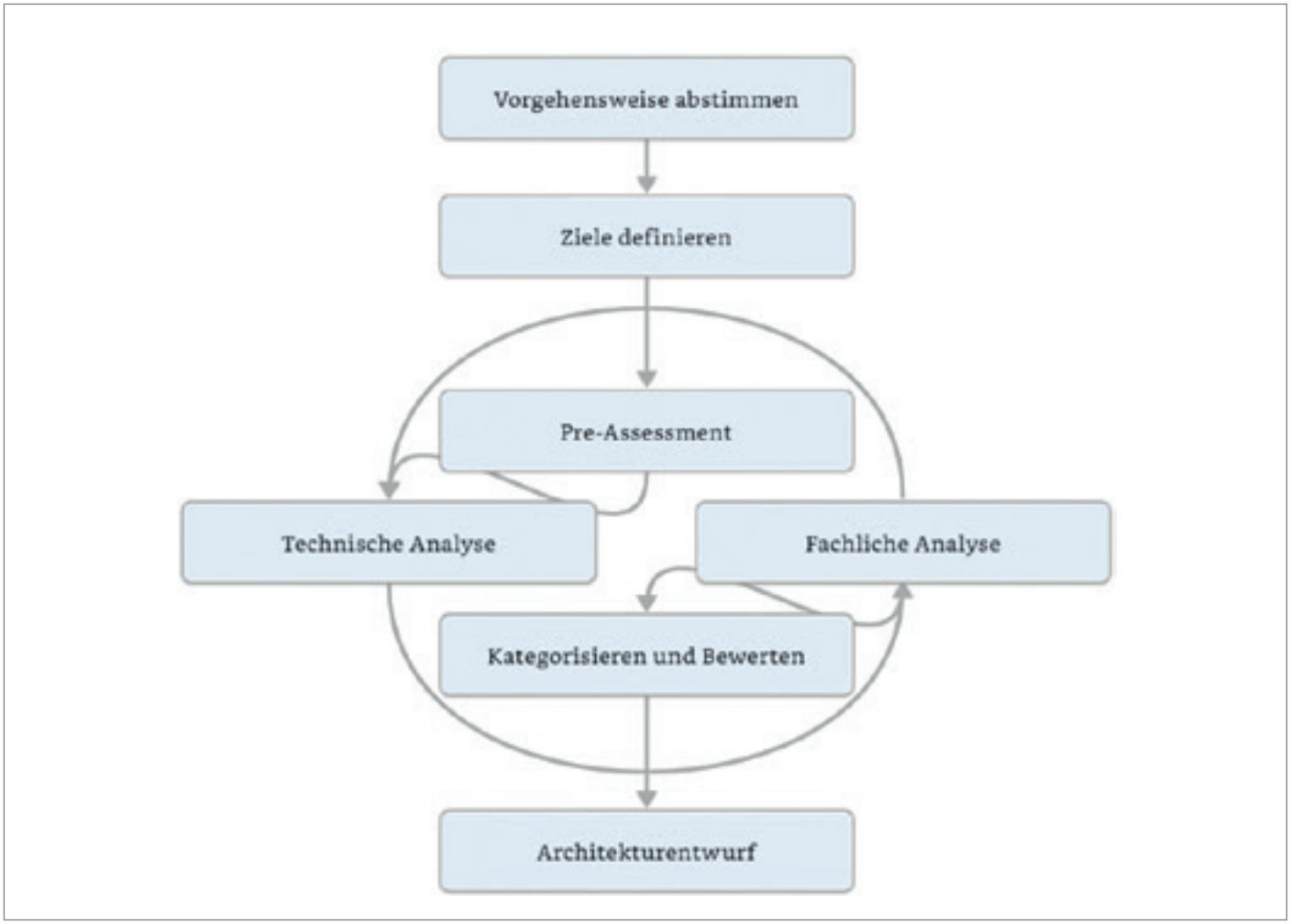

Abbildung 4: Architekturanalyse (©Opitz Consulting)

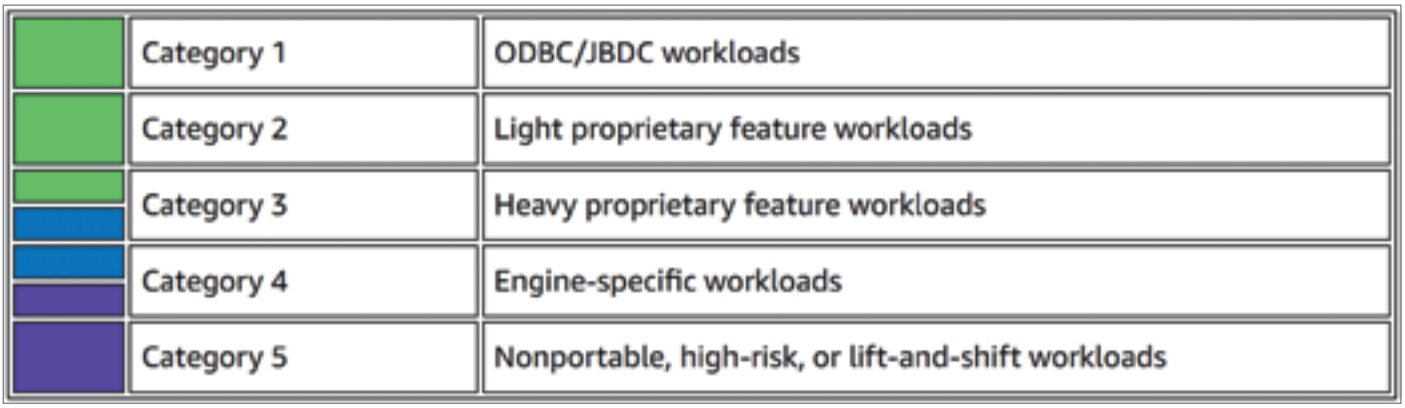

Abbildung 5: AWS Workload Categories (©Amazon Web Services [3])

Wo die Roadmap aufhört – häufig nach dem Pre-Assessment oder der ersten Iteration –, geht es im Laufe der Projektumsetzung weiter ins Detail. Wie schon erwähnt, müssen für die Roadmap nur die wichtigsten Kernpunkte identifiziert werden.

2. Projektplanung

In diesem Abschnitt der Migration geht es darum, Fragen aus dem Pre-Assessment zu analysieren und zu konkretisieren:

- Wieviel Code ist in der DB vorhanden?
- Zu welchem Prozentsatz ist eine automatische Konvertierbarkeit gegeben?
- Welche Codeteile müssen neu geschrieben werden?
- Welche Datentypen und internen Funktionen müssen ersetzt werden?
- Müssen weitere generelle und spezifische Probleme gelöst werden?
- Welche Dokumentation kann hierbei unterstützen?

Bei der Beantwortung dieser Fragen können entsprechende Analysewerkzeuge helfen, etwa das AWS Schema Conversion Tool (SCT).

Analog zu *Abbildung 5* folgt die Klassifizierung der Komplexität (im Folgenden abgekürzt mit „Cat“).

Die Kategorien lassen sich nicht-technisch wie folgt zusammenfassen:

- Cat 1:
  - „simpel, leicht und schnell“
  - „wenig Know-how erforderlich“
- Cat 2:
  - „viele Änderungen geringer Komplexität“
  - „wenige Änderungen hoher Komplexität“
  - „Know-how vorhanden“
- Cat 3:
  - „viele Änderungen hoher Komplexität“
  - „Know-how muss auf-/ausgebaut werden“

Eine ausführlichere Darstellung liefert AWS in der Dokumentation **„Migration strategy for relational databases“** *[3]*. In den Kapiteln **„Qualify workloads“** und **„Choose a migration strategy“** wird auch die zitierte Kategorisierung dargestellt.

Sind die Kategorien „Engine-specific workloads“ (Cat 4) und „Non-portable, high-risk, or lift-and-shift workloads“ (Cat 5) vorhanden, gilt dies als Ausschlusskriterium für eine Migration von Oracle nach PostgreSQL.

Achtung:

Die detaillierte Bewertung dieser Kriterien wird durch das dort ebenfalls zitierte „Workload Qualification Framework“ (WQF) vorgenommen, das nur noch über eine spezielle Anfrage an AWS zu bekommen ist.

Ist dies abgeschlossen, können wir drei grundlegende Fragen beantworten. Diese Fragen sagen uns, ob das Projekt realisierbar ist:

- Ist es technologisch sinnvoll?
- Ist es durchführbar?
- Ist es finanziell attraktiv?

Tipp:

Wir empfehlen dringend, mit einer „Cat 1“-Aufgabe oder einer (bereits sehr gut durchdrungenen) „Cat 2“-Aufgabe zu beginnen. Eine zu komplexe Aufgabe braucht zu viel Abstimmungsaufwand und zieht sich schnell über Monate hin. Damit verspielen Sie nicht nur das Vertrauen des Managements und der beteiligten Stakeholder, sondern auch die Motivation aller Beteiligten.

Die Aufgabe, eine Oracle-Datenbank nach PostgreSQL zu migrieren, *kann,* aber *muss nicht* die Migration der gesamten Datenbank bedeuten. Gerade in einer stark konsolidierten Umgebung, die (um Lizenzkosten zu sparen) viele Anwendungen *ohne* Bezug *zueinander* (!) auf einer leistungsfähigen und verfügbaren Datenbank wie etwa einem Oracle Real Application Cluster vereint, vielleicht sogar auf einer sehr leistungsfähigen Hardware wie einer Exadata, finden sich nicht selten solche voneinander unabhängigen Anwendungen.

Hier kann es sinnvoll sein, wenn möglich, einzelne Anwendungen aus diesem Konstrukt herauszulösen und separat nach PostgreSQL zu migrieren. Das begrenzt die Komplexität in der Gesamtbetrachtung, setzt aber eine sehr gute Analyse im Vorfeld voraus, die sehr wahrscheinlich nicht vom DBA allein geleistet werden kann.

Der DBA kann jedoch viele Details ermitteln und bei der Analyse unterstützen, zum Beispiel durch die Analyse der Datenbankgröße, des Schemas, der Anwendungstabellen sowie von weiteren Punkten, wie

- verwendete Zeichensätze
- problematische Datentypen
- Durchsatz (Redolog-Volumen pro Tag, leider nur bezogen auf die gesamte DB)
- Größe der Large Objects (Performance-Faktor bei der Datenmigration)
- Größe der Indizes (Performance-Faktor bei der Bereitstellung nach der Datenmigration)

3. Umsetzung

Abbildung 6 zeigt den Zyklus, der nun zu Dokumentations- und Testzwecken – vielleicht auch über mehrere Stages mehrfach wiederholt – durchlaufen wird.

Wie bereits erwähnt, sind viele Bereiche einer Migration betroffen. Sie alle müssen an der Umsetzung beteiligt werden. Die folgende Liste zeigt, welche Bereiche hierbei unterstützen können. Je nach Komplexitätskategorie werden mindestens die ersten beiden Bereiche benötigt, gegebenenfalls auch weitere:

- Datenbankadministration (Mitarbeiter:innen mit Expertise zu den Quell- und Zieldatenbanken)
  - Prüfung/Bewertung der Quelle
  - Umsetzung: Schema-Migration
  - Umsetzung: Data-Migration
  - Ausgestaltung der Zielarchitektur
  - Unterstützung bei Performancevergleich und Analyse
- Development (Mitarbeiter:innen mit Anwendungs- und Programmiererfahrung)
  - Codeanpassungen der Applikation
  - Tests
  - Festlegung der Akzeptanzkriterien (mit der Fachabteilung, s. u.)
- Cloud- und Infrastruktur-Architekt:innen
- Netzwerk- und Security-Fachkräfte
- Storage-Fachkräfte
- … und natürlich die Fachabteilung mit Mitarbeiter:innen, die die Inhalte verstehen und Ergebnisse bewerten können.

Damit die Migration nicht zum Selbstzweck gerät, ist es wichtig, sich auch über folgende Punkte Klarheit zu verschaffen:

a. Wie sieht das gewünschte oder erwartete Endergebnis aus? Wann war die Migration für Sie erfolgreich?
b. Wann stoppen Sie das Migrationsvorhaben? Nehmen Sie ein Teilergebnis mit? Schwenken Sie komplett zurück?
c. Welche Risiken haben Sie zu erwarten?
d. Wie können diese Risiken minimiert werden?

In diesem Artikel auf jedes Detail einzugehen, würde den Rahmen sprengen. Eine Übersicht der möglichen Probleme bei einer Migration von Oracle zu PostgreSQL mit SCT enthält die Dokumentation „Oracle to Aurora PostgreSQL Migration Playbook: AWS SCT Action Code Index“ *[4]*. Sie listet zum Beispiel mögliche Probleme auf hinsichtlich

- SQL, Cursors, Merge, Query Hints
- DDL für Tables, (Materialized) Views, Triggers, Data Types
- Transaction Isolation, Stored Procedures, Partitioning, DB Links

Ein sehr triviales Beispiel mit möglicherweise großer Wirkung ist die unterschiedliche Behandlung von NULL, die im EnterpriseDB-Artikel *„How NULL and empty strings are treated in PostgreSQL vs Oracle“ [5]* dargestellt ist.

Durch die unterschiedliche Behandlung von NULL kann es vorkommen, dass eine andere als die erwartete Ergebnismenge zurückgeliefert wird. Der Grund: Oracle setzt einen String mit 0 Zeichen ('') mit NULL gleich, während PostgreSQL diesen als „leer“ und ungleich NULL bewertet. Letzteres entspricht dem ANSI/ISO-SQL-Standard.

Wenn Oracle-Kompatibilität bei möglichst wenig Code-Anpassungen gewünscht ist, kann EnterpriseDB eine gute Wahl sein. EnterpriseDB vertreibt neben Tools für das freie PostgreSQL auch eine eigene PostgreSQL-Variante, die Oracle Features und Packages wie zum Beispiel UTL_FILE, UTL_MAIL oder DBMS_* Packages enthält.

Um einen „Hersteller-Support“ wie bei Oracle zu bekommen, bietet EnterpriseDB zudem einen professionellen Troubleshooting- und insbesondere auch Break/Fix-Support sämtlicher PostgreSQL-Varianten, Tools und Extensions durch eigene PostgreSQL-Entwickler.

Zur Bewertung analog SCT verwendet EnterpriseDB ein Webportal, das EDB Migration Portal *[6]*.

Diese beiden Blog-Artikel enthalten umfängliche Darstellungen von Unterschieden und Herausforderungen bei der Migration von Oracle zu PostgreSQL:

- *EnterpriseDB: „The Complete Oracle to Postgres Migration Guide: Move and Convert Schema, Applications and Data, EnterpriseDB“ [7]*
- *AWS: „Challenges When Migrating from Oracle to PostgreSQL—and How to Overcome Them“ [8].*

All diese kleinen und größeren Unterschiede sollten idealerweise bekannt und berücksichtigt worden sein.

Da meist nicht ausreichend viel Zeit zur Verfügung steht, brauchen Sie einen Mix aus kenntnisreicher Unterstützung, genauem Testen und sachgerechter Prüfung.

Fazit

Zum Schluss noch einmal der wichtige Hinweis:

„Testen, testen, testen! Lieber einmal zu viel getestet als einmal zu wenig.“

Denn ist die Applikation einmal umgeschaltet und zur Verwendung freigegeben worden, kann es schwierig werden, die in PostgreSQL geänderten Daten zurück nach Oracle zu spielen.

Wer im Vorfeld auf die Mahnung und den Vorschlag gehört hat, die Daten aus der Zielumgebung mittels AWS Database Migration Service, Oracle GoldenGate, Quest Shareplex oder anderer Methoden zurück in eine Kopie der Oracle-Datenbank zu replizieren, kann bei Bedarf mit bestenfalls kurzer Unterbrechung auf die Quelldatenbank zurückschalten, ohne neue Daten zu verlieren.

Wer mehr erfahren möchte kann im DOAG-Archiv weitere Beiträge von mir zu diesem Thema finden:

- DOAG-Webinar: „Aufgezeichnet: Migration Factory – Oracle to AWS RDS for PostgreSQL“ [9], bei dem ich näher auf die Bedienung von SCT eingehe
- Vortrag auf der DOAG-Konferenz 2022: *„Migrationen von Oracle (on-premises) nach PostgreSQL und Oracle RDS in der Cloud“ [10]*

Viel Spaß und Erfolg bei Ihrer Migration!

Quellen

[1] Star-Methode, Wikipedia, *https://de.wikipedia.org/wiki/Star-Methode*

[2] Andrea Windolph (2022): Roadmapping: Was und wofür? Ein kompakter Überblick, projekte-leicht-gemacht.de, *https://projekte-leicht-gemacht.de/blog/business-wissen/roadmapping/* Kapitel: „Unterschied zwischen Roadmapping und Projektplanung“ und Bildreferenz: *https://projekte-leicht-gemacht.de/blog/*

business-wissen/roadmap-beispiele/#vorlage
[3] Amazon Web Services AWS (2020): AWS Prescriptive Guidance – Migration strategy for relational databases, *https://docs.aws.amazon.com/prescriptive-guidance/latest/strategy-database-migration/qualify-workloads.html*
[4] Amazon Web Services AWS: Oracle to Aurora PostgreSQL Migration Playbook, *https://docs.aws.amazon.com/dms/latest/oracle-to-aurora-postgresql-migration-playbook/chap-oracle-aurora-pg.tools.actioncode.html*
[5] Thom Brown (2023): How NULL and empty strings are treated in PostgreSQL vs Oracle, EnterpriseDB, *https://databaserookies.wordpress.com/2023/01/09/substr-functionality-differences-between-oracle-and-postgresql-what-you-need-to-know/*
[6] EDB Migration Portal, EnterpriseDB, *https://migration.enterprisedb.com/*
[7] Raghavendra Rao (2020): The Complete Oracle to Postgres Migration Guide: Move and Convert Schema, Applications and Data, EnterpriseDB, *https://edb.prod.acquia-sites.com/blog/the-complete-oracle-to-postgresql-migration-guide-tutorial-move-convert-database-oracle-alternative*
[8] Silvia Doomra (2018): Challenges When Migrating from Oracle to PostgreSQL—and How to Overcome Them, Amazon Web Services AWS, *https://aws.amazon.com/de/blogs/database/challenges-when-migrating-from-oracle-to-postgresql-and-how-to-overcome-them/*
[9] Christian Ballweg (2020), Aufgezeichnet: Migration Factory – Oracle to AWS RDS for PostgreSQL, DOAG, *https://www.doag.org/de/home/news/aufgezeichnet-migration-factory-oracle-to-aws-rds-for-postgresql*
[10] Christian Ballweg (2020), Migrationen von Oracle (on-premises) nach PostgreSQL und Oracle RDS in der Cloud, DOAG, *https://en.shop.doag.org/download.1db52e-a32027415831e77c86c82f1592_1/*

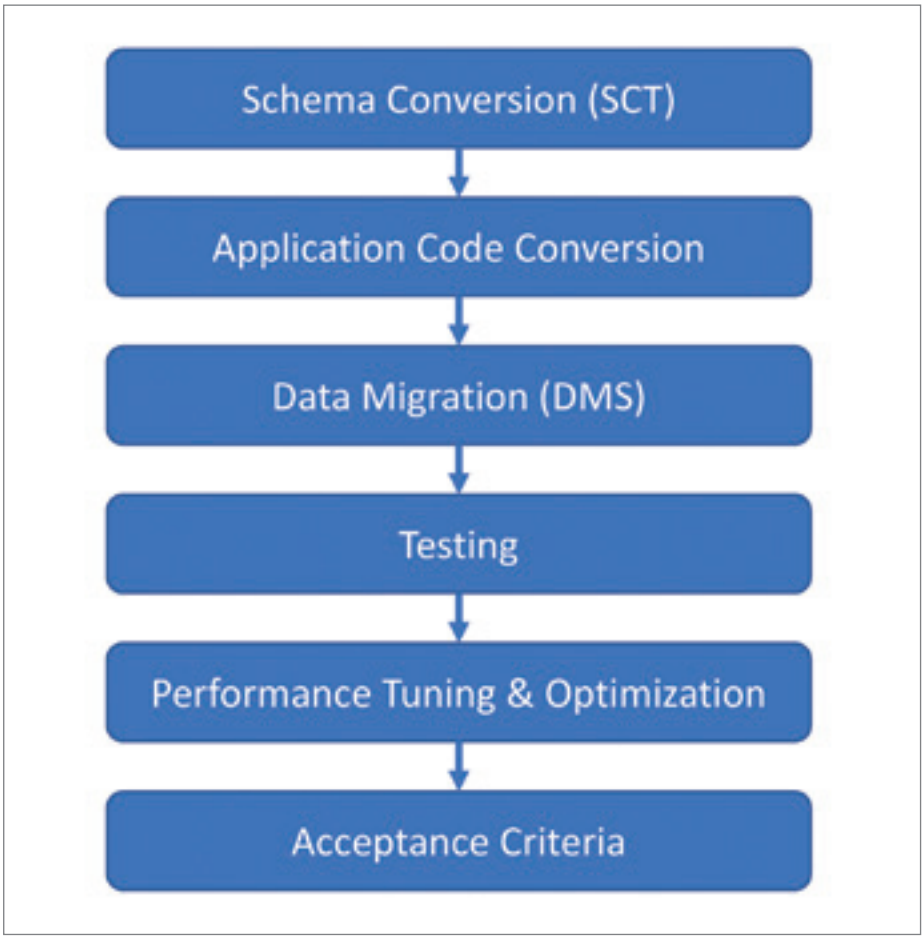

Abbildung 6: Migrationszyklus (Quelle: Christian Ballweg)

Über den Autor

Christian Ballweg ist seit mehr als 10 Jahren als Solution Architect bei der Opitz Consulting Deutschland GmbH beschäftigt. Seine über 20 Jahre Erfahrung in der IT erwarb er im Bereich Design, Aufbau und Betrieb von Oracle-Datenbankumgebungen mit den Schwerpunkten Data Warehouse, Hochverfügbarkeit, Performance Tuning, Upgrades und Minimal-Downtime-Migrationen sowie bei Konsolidierungs- und Lizenzoptimierungsprojekten. Er liebt es, sein Wissen als Berater, als Referent bei Fachkonferenzen oder als Autor von Fachartikeln weiterzugeben.

Christian Ballweg
Christian.Ballweg@opitz-consulting.com