Amazon.comにおける OracleからPostgreSQLへの移行

アマゾン ウェブ サービス ジャパン株式会社
データベース スペシャリスト ソリューション アーキテクト
新久保 浩二

アンケートにご協力ください。

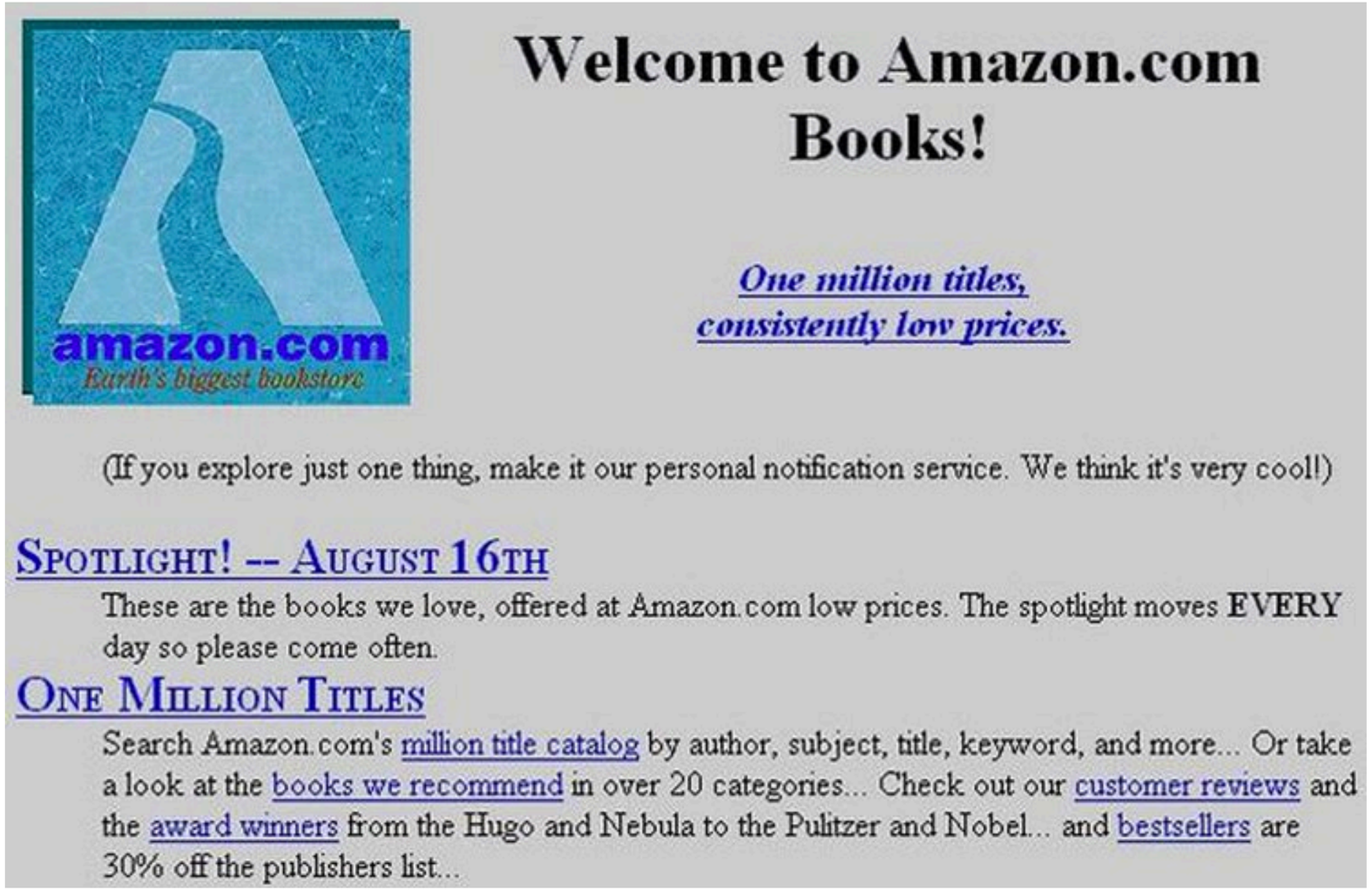
https://bit.ly/2S6VTUY

アジェンダ

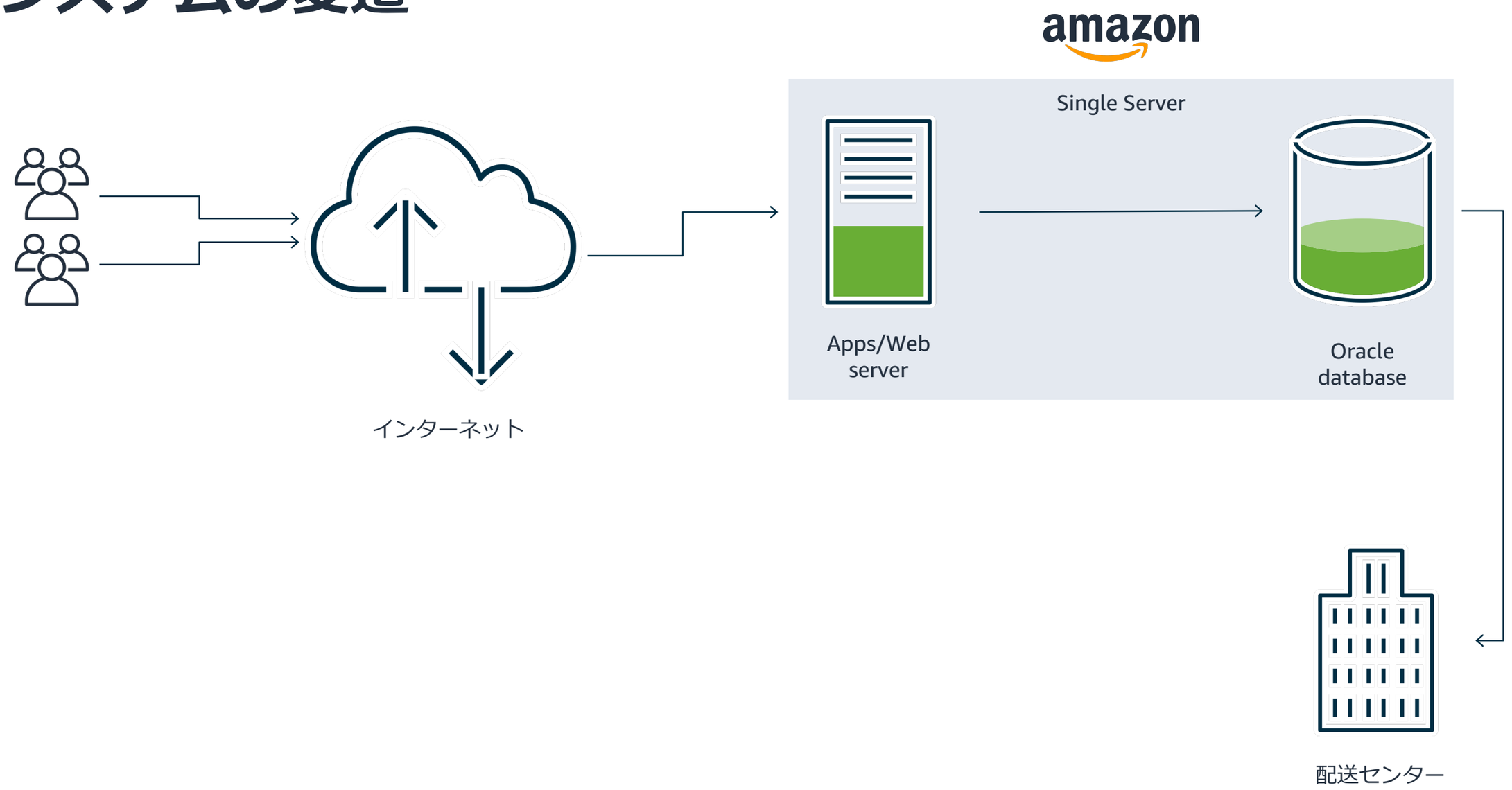
- はじめに
    - Amazonのレガシー データベース基盤
- データベースを移行した理由
- 移行戦略と学んだ教訓
- 移行の効果
- OracleからPostgreSQLへ移行する場合の技術的なポイント

Amazonのレガシー データベース基盤

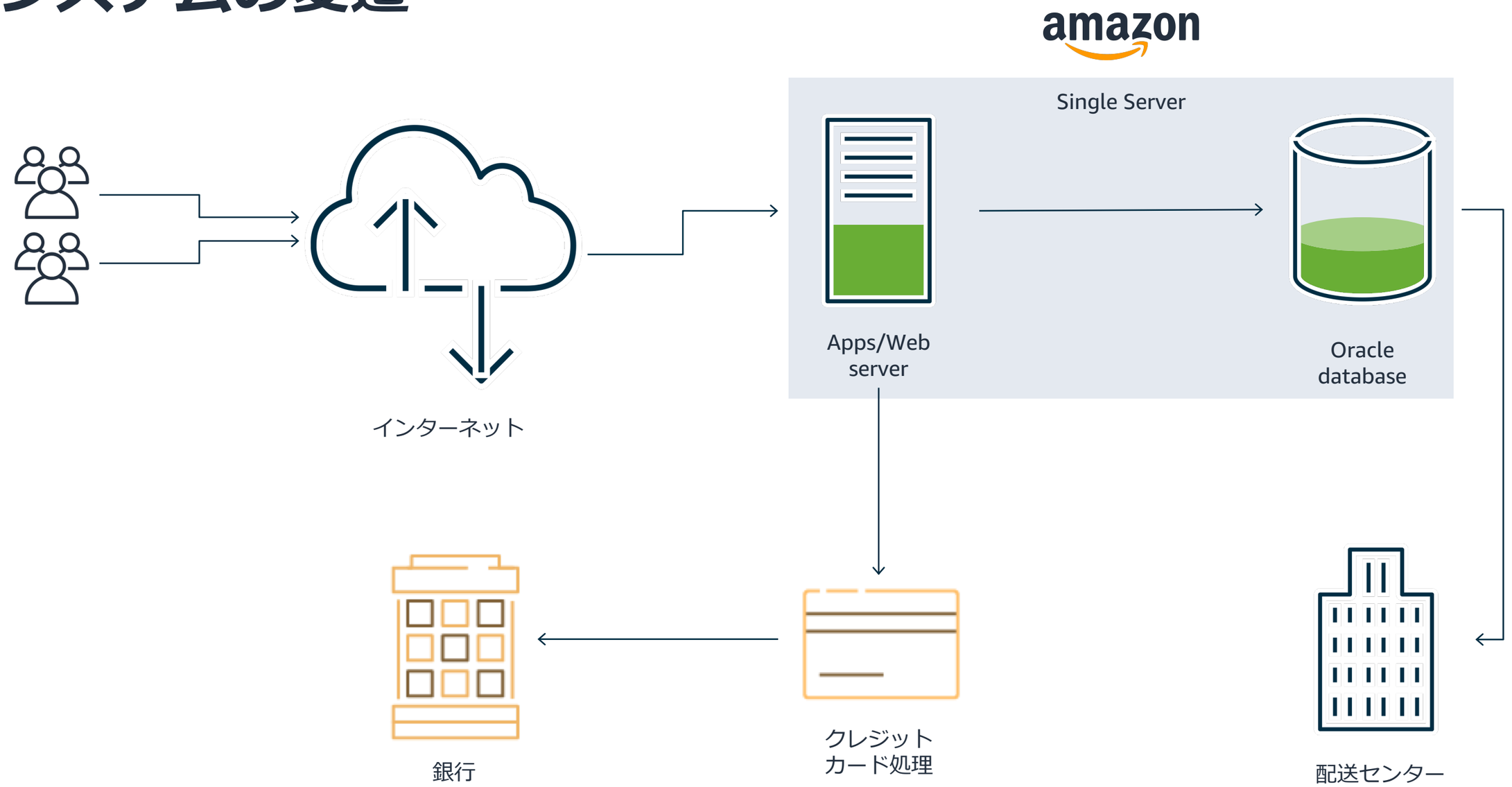
はじめに



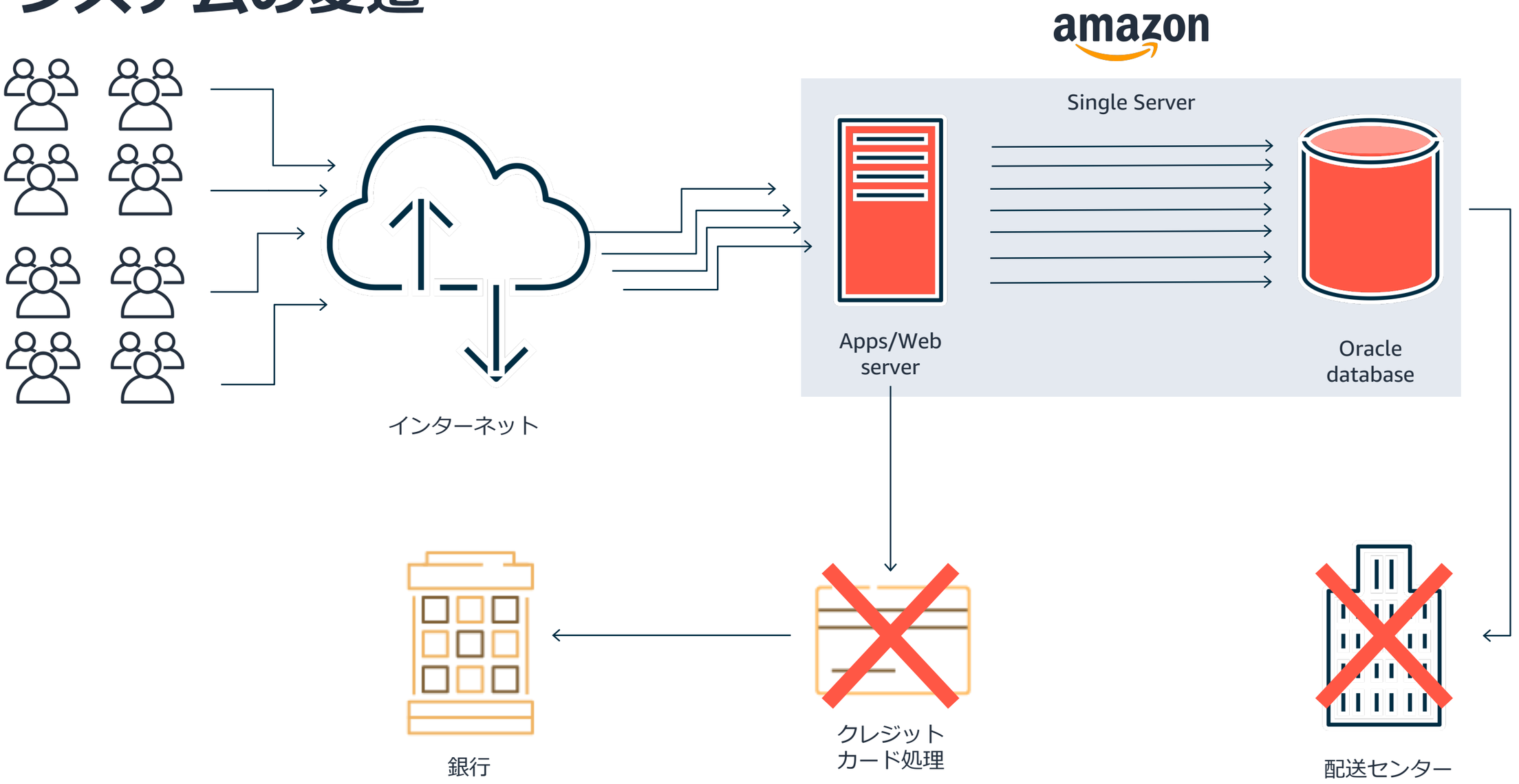
システムの変遷



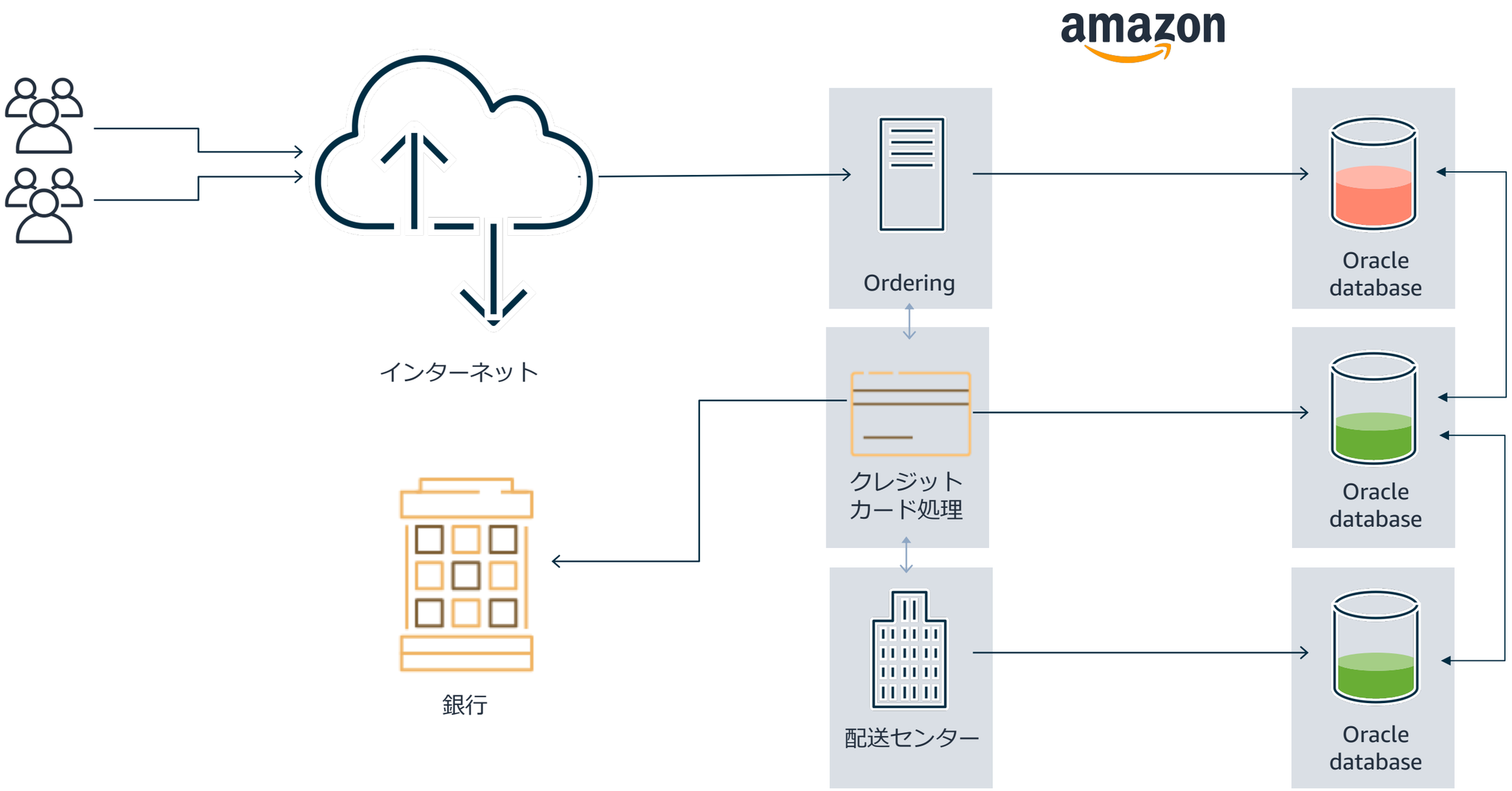
システムの変遷



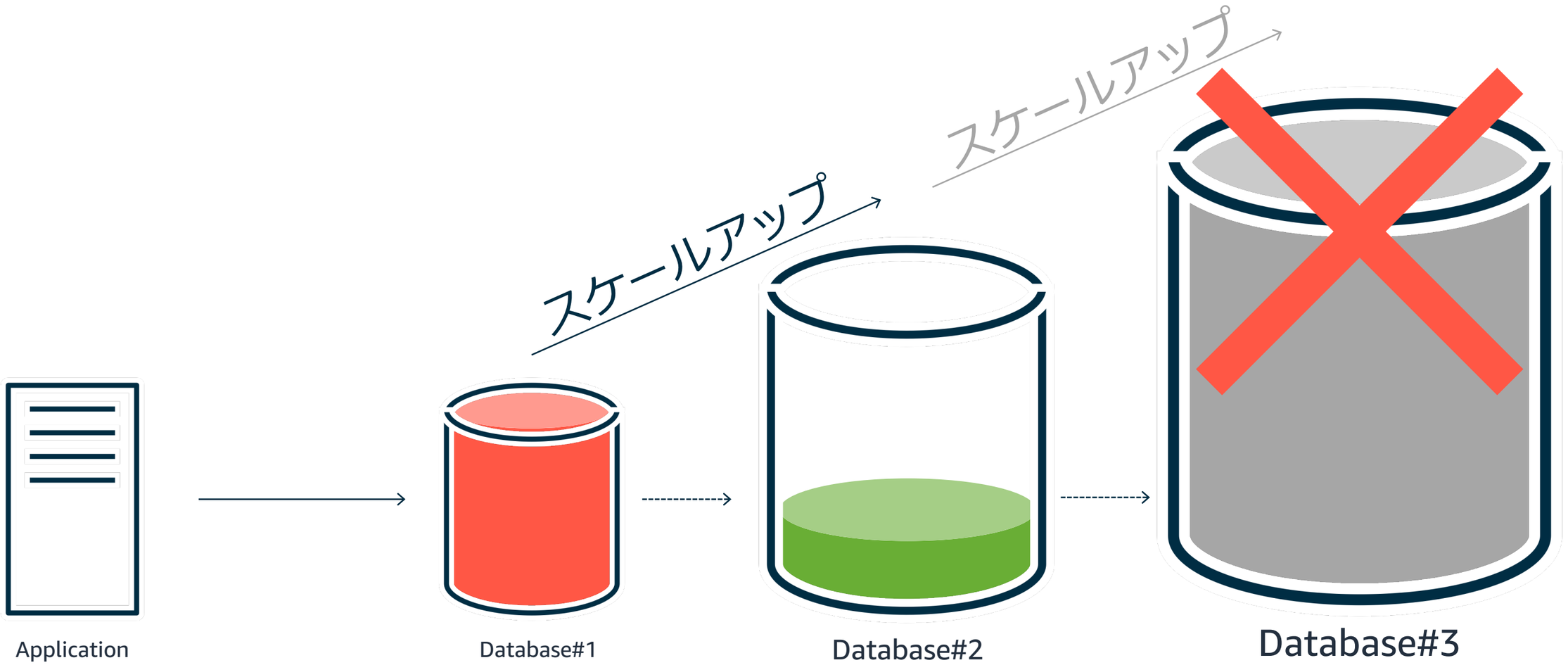
システムの変遷



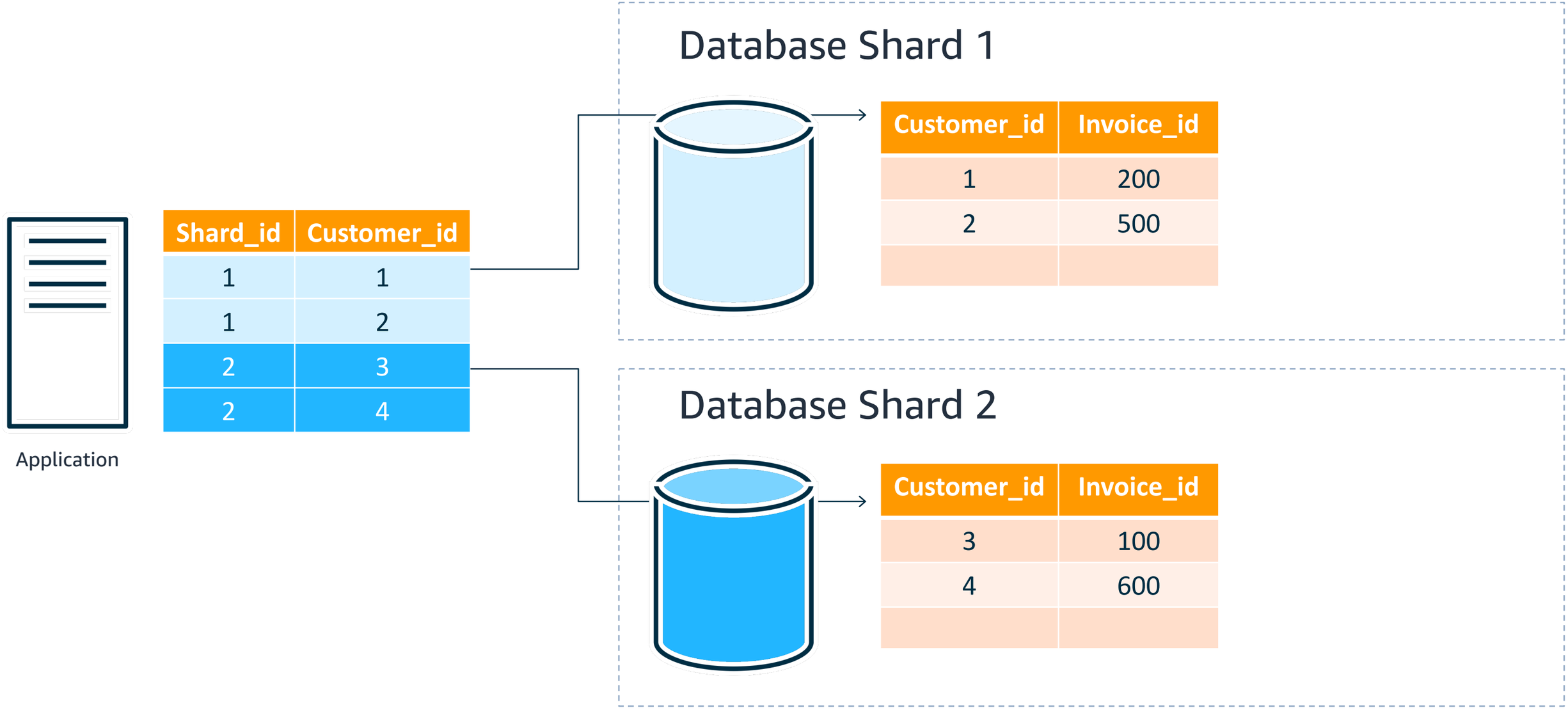
システムの変遷



データベースのスケーラビリティの問題



データベース シャーディング



2017年時点のデータベース基盤

OLTPデータベース

- 総データ量 75PB
- 約 7,500 OLTP データベース
- 100以上のチームが数千のアプリケーションで利用
- 世界中で数百万の受発注、決済等の処理

2017年時点のデータベース基盤

Data Warehouseデータベース

- 総データ量 50PB
- テーブル数 75,000
- 600,000 分析ジョブ /日
- 1,800 チームがデータを公開 /日
- 3,300 チームがデータを利用 /日

OLTPデータベース

- 総データ量 75PB
- 約 7,500 OLTP データベース
- 100以上のチームが数千のアプリケーションで利用
- 世界中で数百万の受発注、決済等の処理

データベースを移行した理由

移行要件

サービス可用性

100%に近い可用性
と必要に応じて
スケール出来ること

データ可用性

すべてのデータが
常に利用可能で
一貫性があること

移行時間

移行期間でも
ダウンタイムなし、
もしくは
わずかなダウンタイム
で完了すること

移行を決めた理由

1. データ量の増大とグローバル市場の拡大によるスケーラビリティのリスク
2. データ量とトランザクションの増大によるレイテンシーのリスク
3. HW / SW コストのリスク
4. レガシーなコード / アーキテクチャによる可用性のリスク
5. HWをプロビジョニング・管理する事による運用管理のリスク

スケールし続けるシステムの管理

- プライムデーに対応するためのシステム拡張作業に60 週間かかるチームもあった
- ベンダーの手厚いサポートを受けてもスケール時の障害リスクが高かった
- 顧客に優しくない懲罰的なライセンス戦略

トランザクションレート

ビジネスの拡大につれて
増え続けるトランザクシヨン

レイテンシーを向上させ運用管理コストを削減

- Amazonのビジネスの成長スピードに負けない応答スピードの確保が困難だった
- 指数関数的に増え続けるデータの管理にコストが非常にかかっていた

レイテンシー

急激に増え続ける膨大なデータ
に対するレイテンシーの向上

HW / SW コストのリスク

- HWの価格と可用性とのバランスを取ることが難しく、ビジネスのピークとHWの調達のタイミングを合わせづらい
- SWの価格設定とベンダーの戦略への対応はコストが高く管理が困難
  - リニューアル、HWによって変わる、仮想化に対するライセンス、キャパシティプランニング・・・

CAPEX

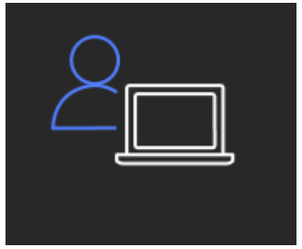
HW/SW コスト

Software

ライセンス管理

可用性と運用管理のリスク

- データベースは拡張と縮退、両方必要
- レガシーシステムの拡張/縮退、運用管理に数百人
- オペレーションは確実・予測可能・計測可能であること
- データは常にアクセスできること
- データは常に一貫性を保つこと



可用性と運用

HW プロビジョニングと
運用管理

プライムデーによりさらなるスケーラビリティが必要に

プライムデーによりピークは1年に1回から半年に

- 1回のシステム拡張で許容される時間は半分に

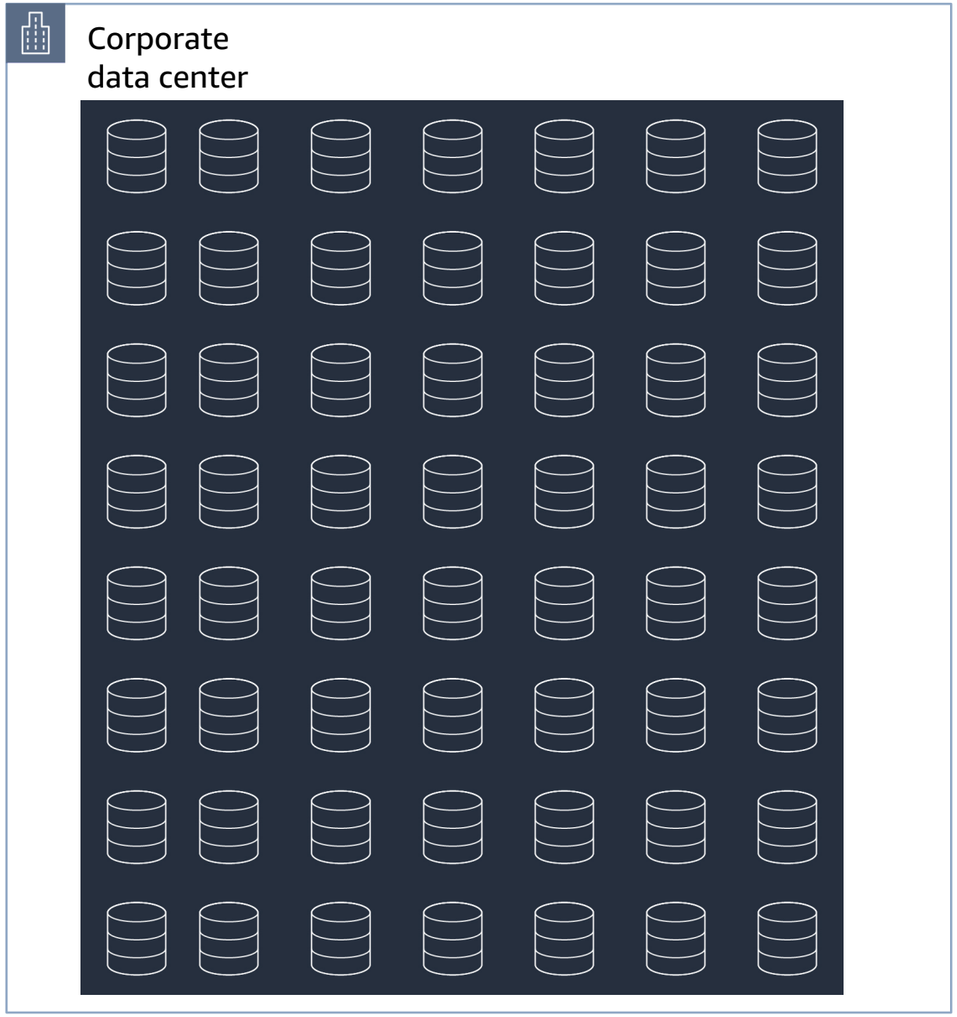
プライムデー自体の規模の拡大

- 2015年 9カ国で3,440万点の注文
- 2019年 18カ国で1億7500万点の注文 (Amazonの歴史上、最大のイベント)

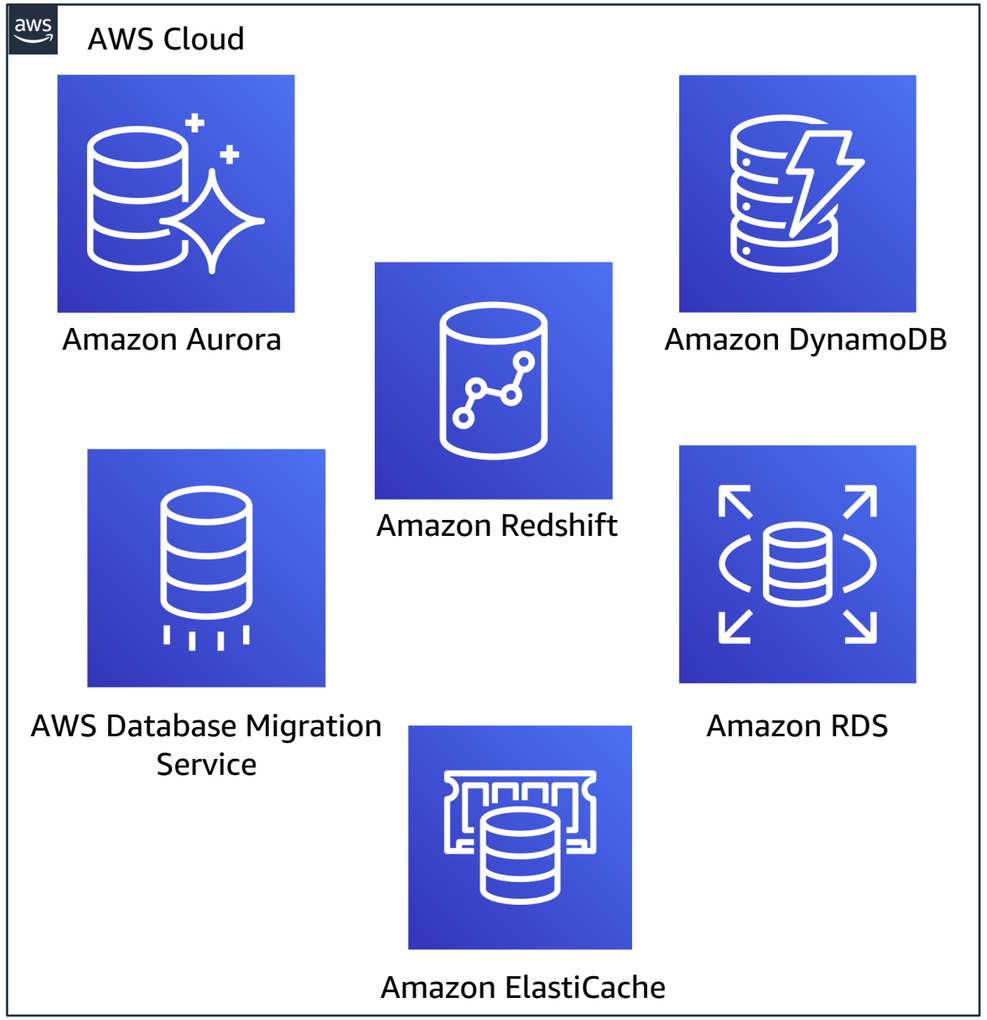
移行戦略と学んだ教訓

移行方針

Oracleデータベース



AWS purpose-builtデータベース



戦略① 可視性の向上 (1/3)

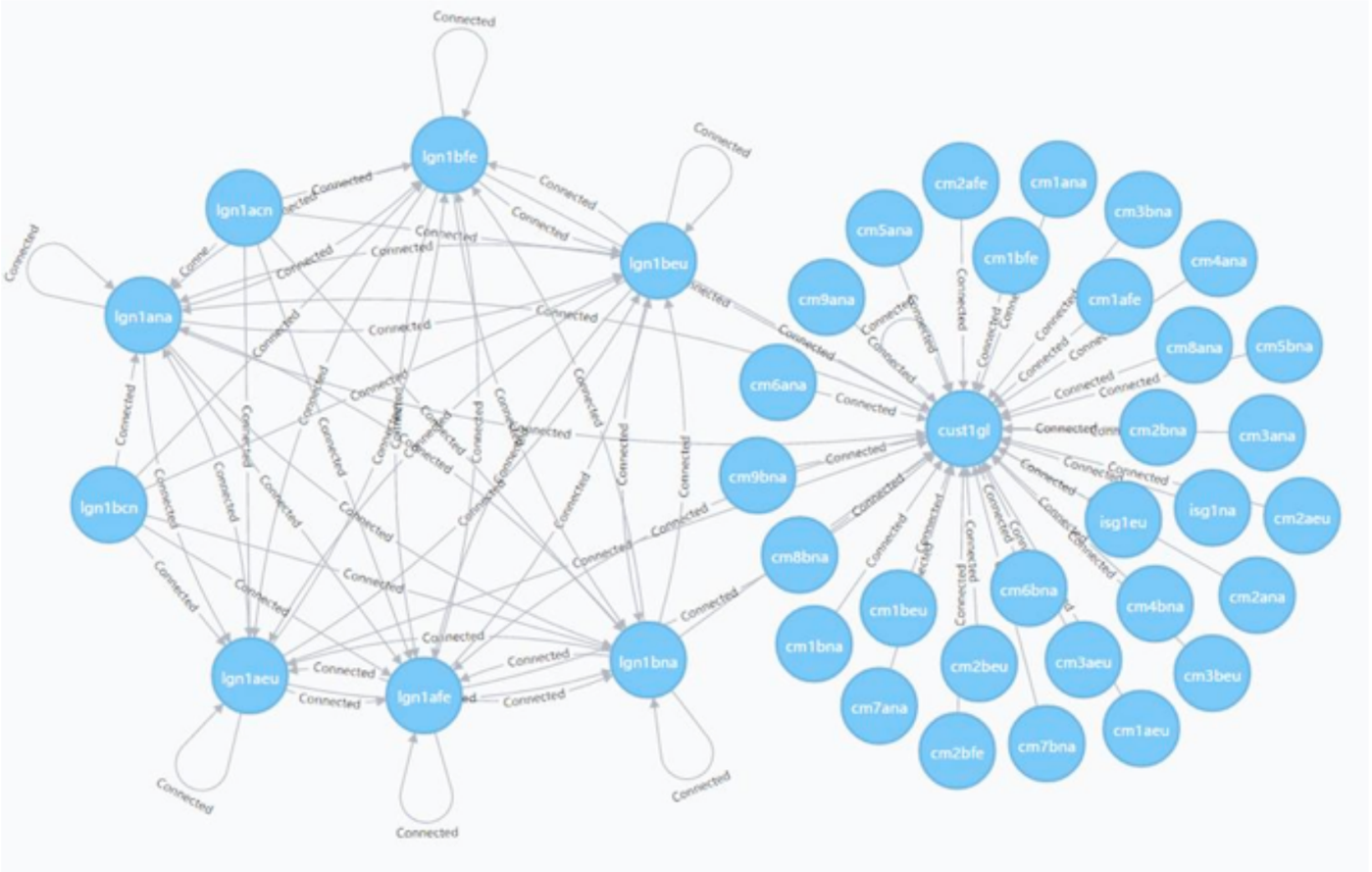
適切なKPIや成功の評価基準の設定による移行の進捗管理

- 現在のOracleデータベースの数
- 新たに実装されたOracleデータベースの数
- 移行された(廃止された)Oracleデータベースの数
- データベースの管理者名

戦略① 可視性の向上 (1/3)

適切なKPIや成功の評価基準の設定による移行の進捗管理

- 現在のOracleデータベースの数
- 新たに実装されたOracleデータベースの数
- 移行された(廃止された)Oracleデータベースの数
- データベースの管理者名

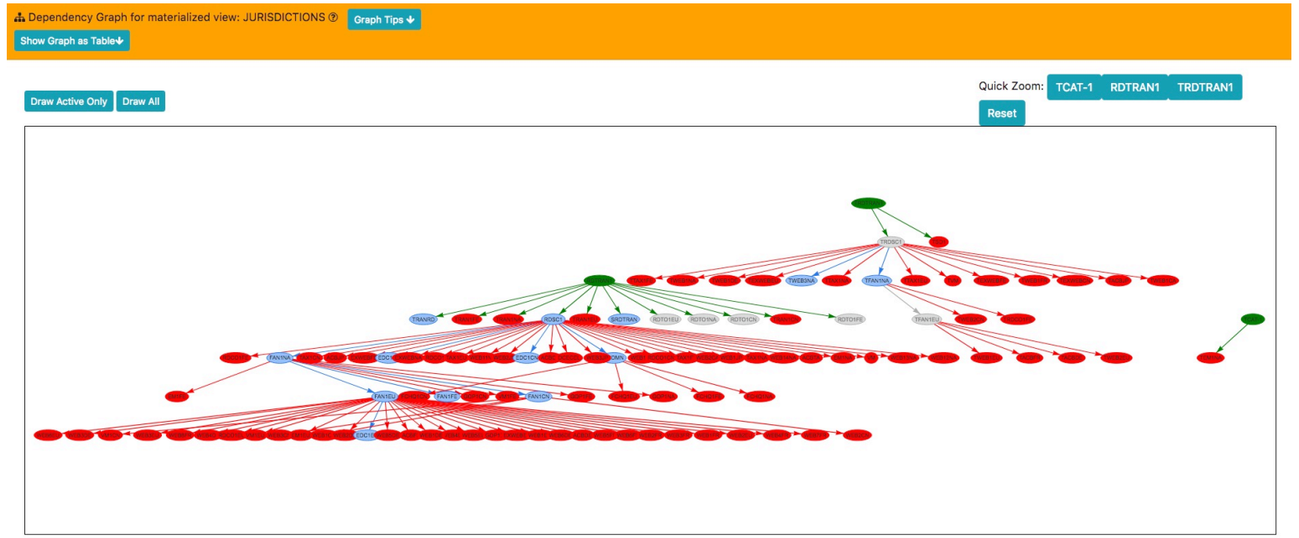


戦略① 可視性の向上 (2/3)

- Oracleデータベースのインベントリ情報
- データベースモジュールの依存関係
- CRUDレポート
    - Create Read Update Delete
- データベースオブジェクトのOwner/Type
- DWジョブとDB接続情報
- Advanced Search 機能
- DB移行のバーンダウンチャート
- プロジェクトステータス
- データベース移行のステータス

戦略① 可視性の向上 (2/3)

- Oracleデータベースのインベントリ情報
- データベースモジュールの依存関係
- CRUDレポート
    - Create Read Update Delete
- データベースオブジェクトのOwner/Type
- DWジョブとDB接続情報
- Advanced Search 機能
- DB移行のバーンダウンチャート
- プロジェクトステータス
- データベース移行のステータス



戦略① 可視性の向上 (3/3)

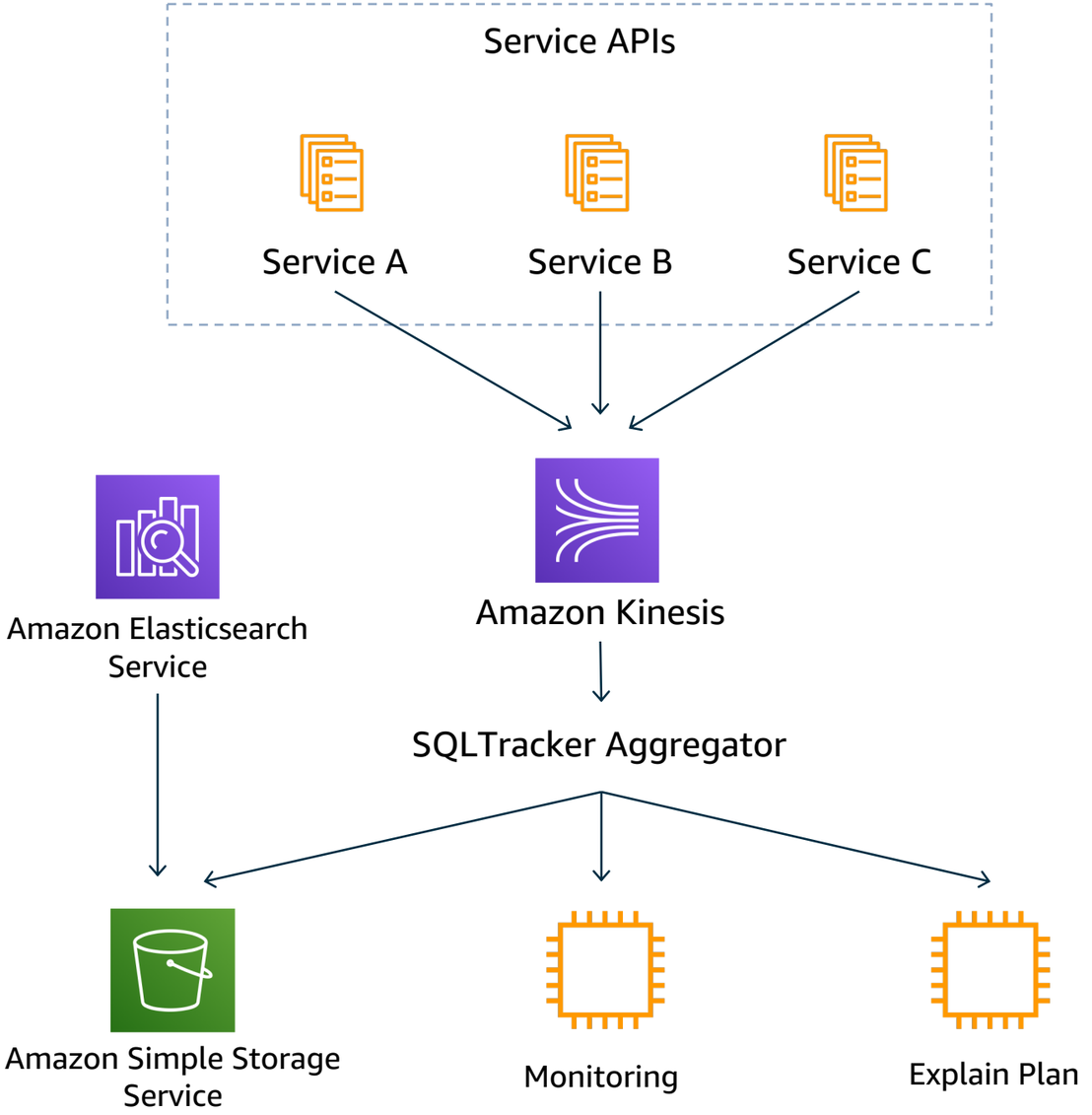
SQL Tracker

- 実行された全てのSQL文をトラックし分析するためのJavaライブラリ
- java.sql.Connection / javax.sql.DataSource インスタンスのWrapper

戦略① 可視性の向上 (3/3)

SQL Tracker

- 実行された全てのSQL文をトラックし分析するためのJavaライブラリ
- java.sql.Connection / javax.sql.DataSource インスタンスのWrapper

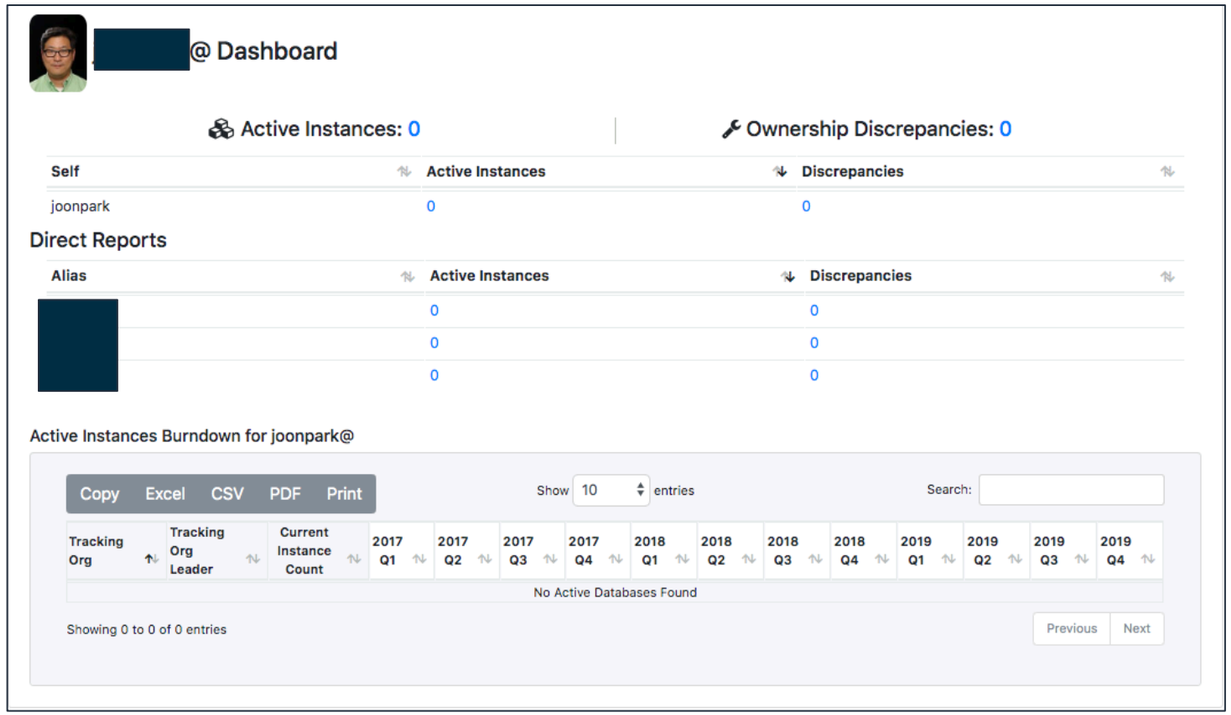


戦略② 経営陣のサポートを加速 (1/2)

- 経営幹部のサポート
- AWSとのパートナーシップ
  - アカウントチーム
  - 各サービスのスペシャリスト
    - Aurora/RDS, DynamoDB, DMS, Redshift
- 社内のAWS推進派や新しい手法に興味がある人達
- 進捗を把握し、説明責任を果たすためのレポート体制

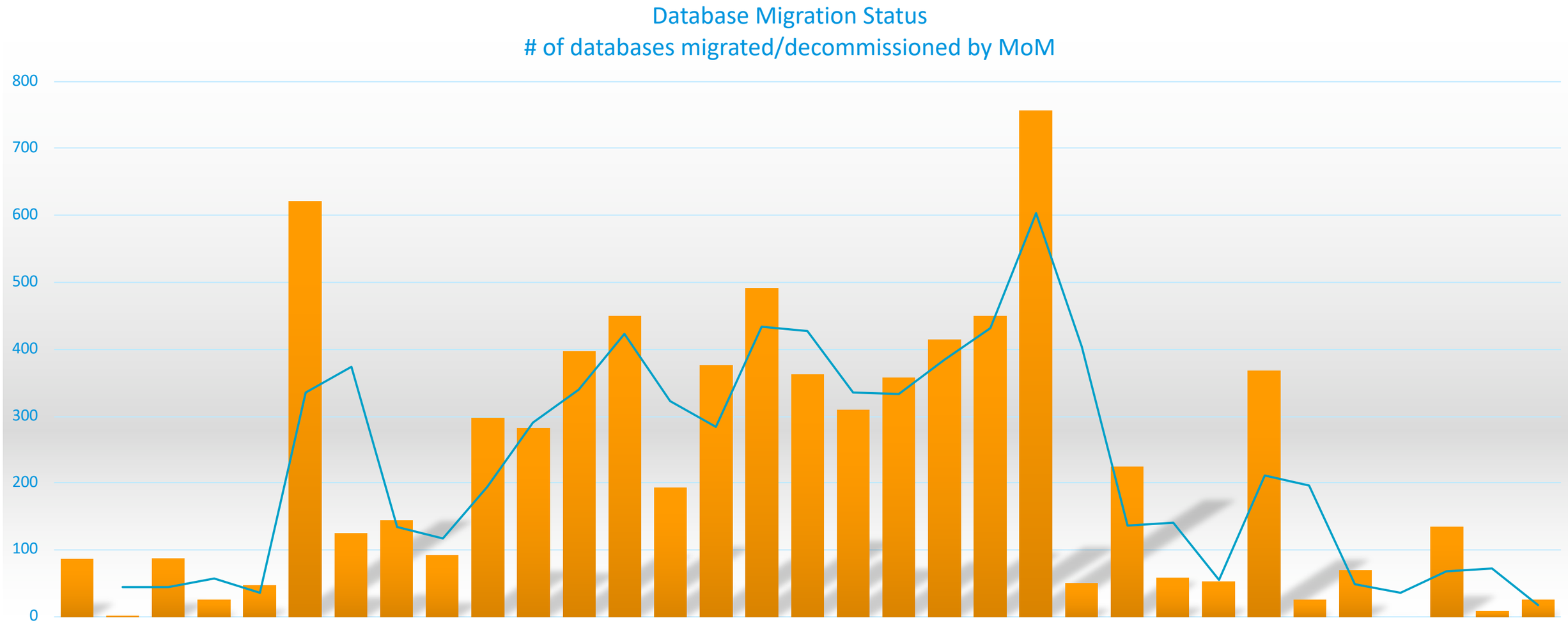
戦略② 経営陣のサポートを加速 (1/2)

- 経営幹部のサポート
- AWSとのパートナーシップ
  - アカウントチーム
  - 各サービスのスペシャリスト
    - Aurora/RDS, DynamoDB, DMS, Redshift
- 社内のAWS推進派や新しい手法に興味がある人達
- 進捗を把握し、説明責任を果たすためのレポート体制



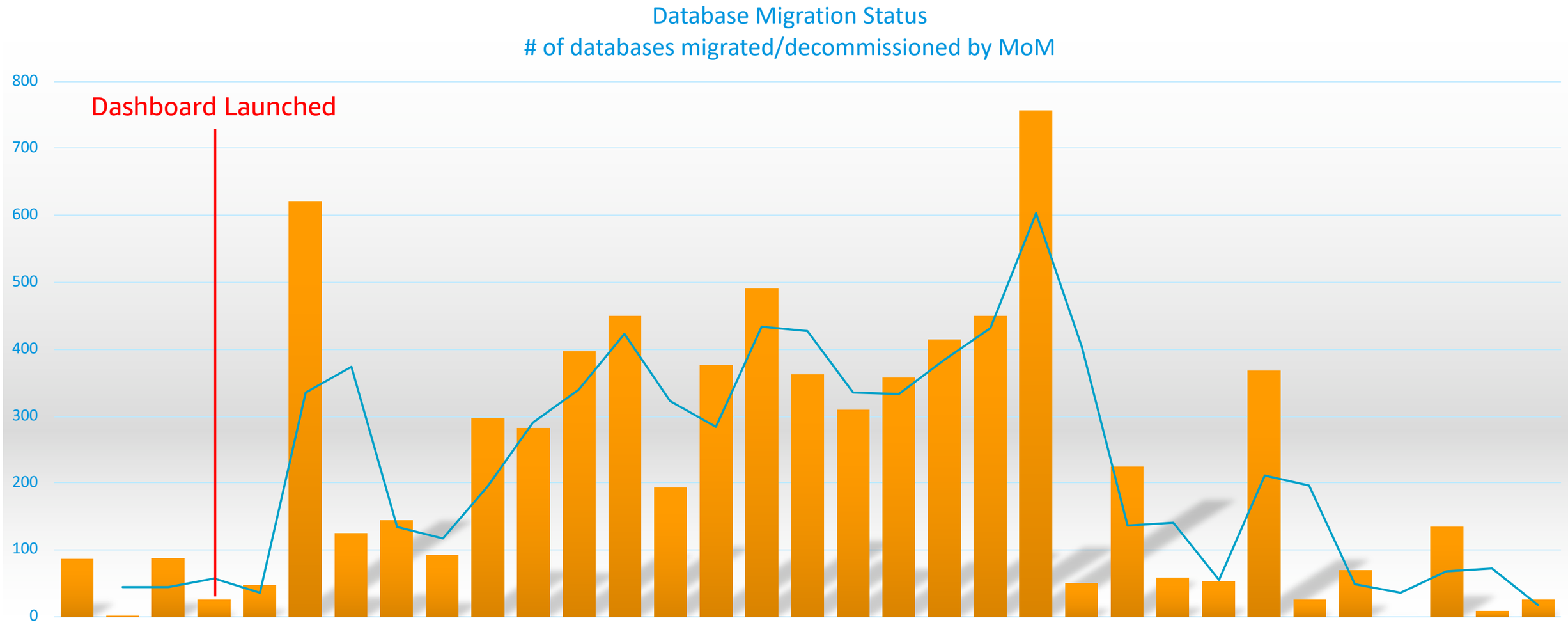
戦略② 経営陣のサポートを加速 (2/2)

データベース移行の進捗を可視化



戦略② 経営陣のサポートを加速 (2/2)

データベース移行の進捗を可視化



戦略③ FUDと実際の技術的課題に対処

- 社内Oracle DBAからの反発があった
  - 移行に対する疑念や恐れを表明
  - ⇔ 実際の技術的課題とは違う
- 移行することによって得られるメリットを考える
- 20年以上もの間、データベースの移行は行われてこなかった
- 適応するためには時間が必要なことを理解する

Fear Uncertainly Doubt
(恐れ 不確実性 疑念)

vs

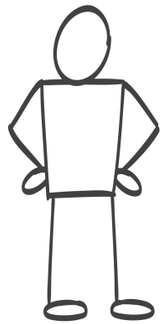
実際の技術的課題

Oracle DBAからの支援を得る

Oracle DBA からの支援を得ることで、
リスクの緩和、移行における
依存関係を明確にすることが可能

Oracle DBAに新たなキャリアパスと
トレーニングの機会を提供

- データベースエンジニア
- ソリューションアーキテクト
- システム開発エンジニア
- DB移行スペシャリスト
- AWS認定資格者
- ・・・



Oracle
DBA

Oracle DBAからの支援を得る

Oracle DBA からの支援を得ることで、
リスクの緩和、移行における
依存関係を明確にすることが可能

Oracle DBAに新たなキャリアパスと
トレーニングの機会を提供

- データベースエンジニア
- ソリューションアーキテクト
- システム開発エンジニア
- DB移行スペシャリスト
- AWS認定資格者
- ・・・

Oracle DBA

Oracle DB Migrations

戦略④ フォースマルチプライヤー

- フォースマルチプライヤー
    - 情報共有する事により、スキルを持ったエンジニアを増やしていく

- AWSの定期的な勉強会の実施
    - Aurora/RDS, Redshift, DynamoDB, DMS・・・

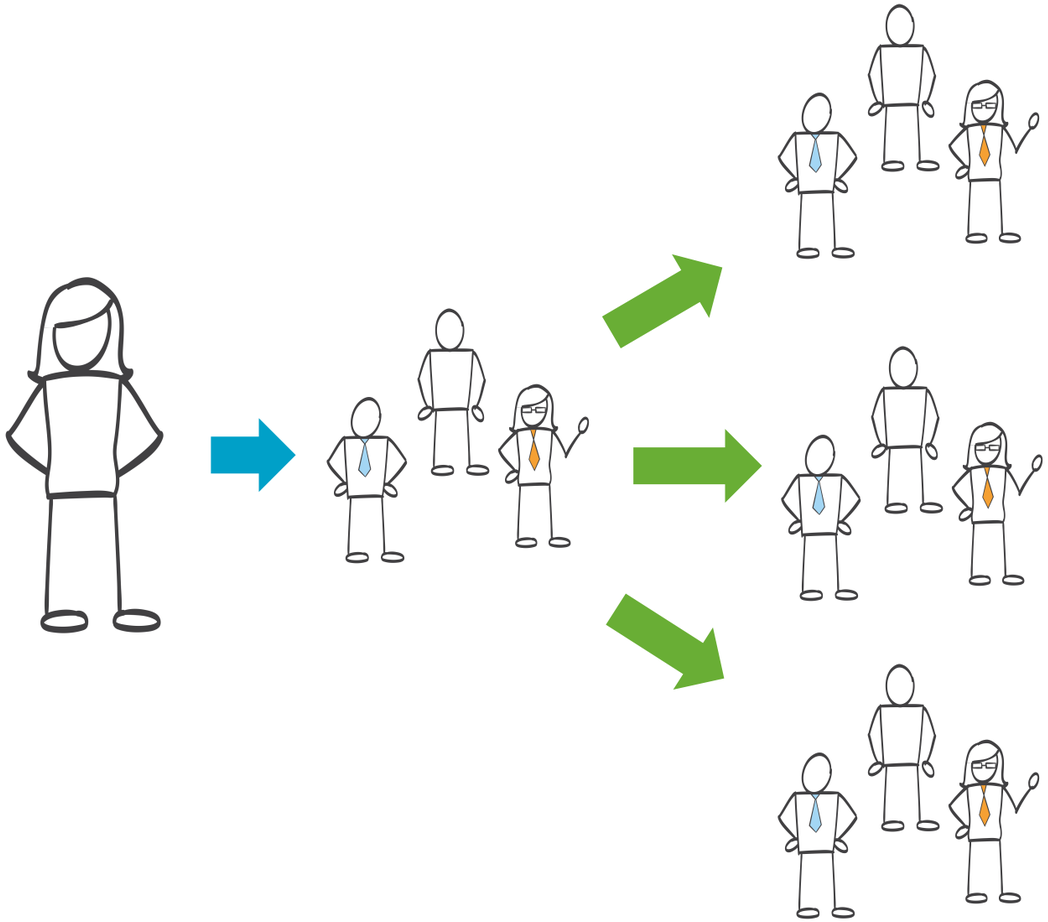
- リファレンスアーキテクチャ、ベストプラクティス、デザインパターン、Tipsの共有

戦略④ フォースマルチプライヤー

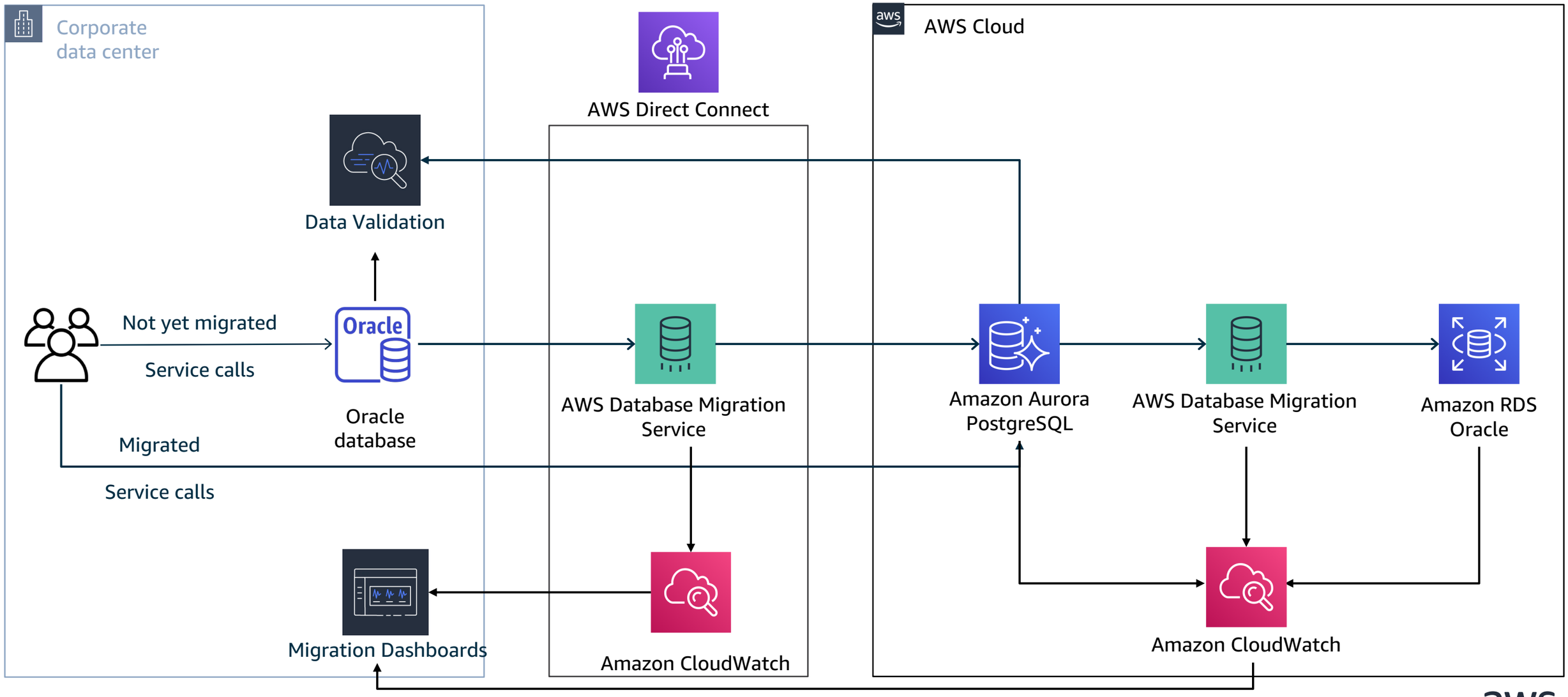
- フォースマルチプライヤー
    - 情報共有する事により、スキルを持ったエンジニアを増やしていく

- AWSの定期的な勉強会の実施
    - Aurora/RDS, Redshift, DynamoDB, DMS・・・

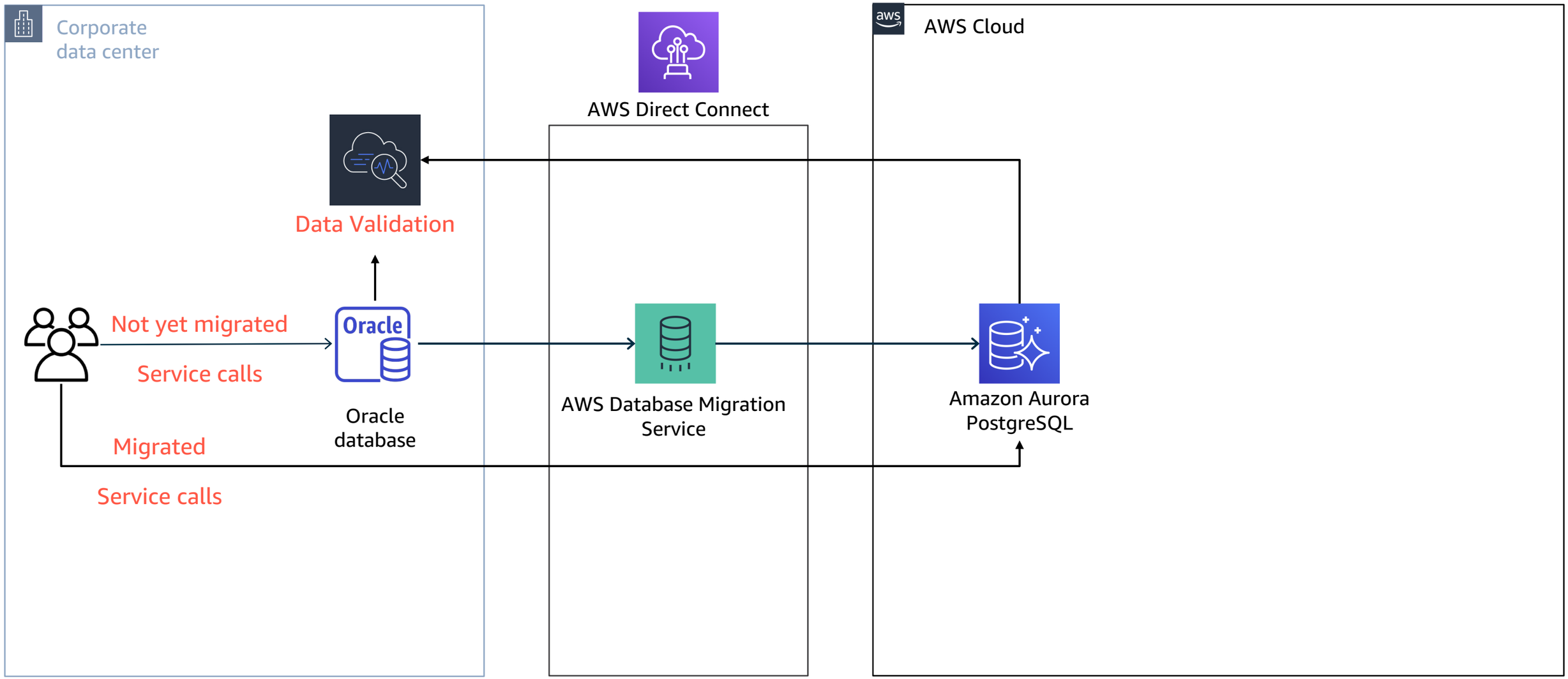
- リファレンスアーキテクチャ、ベストプラクティス、デザインパターン、Tipsの共有



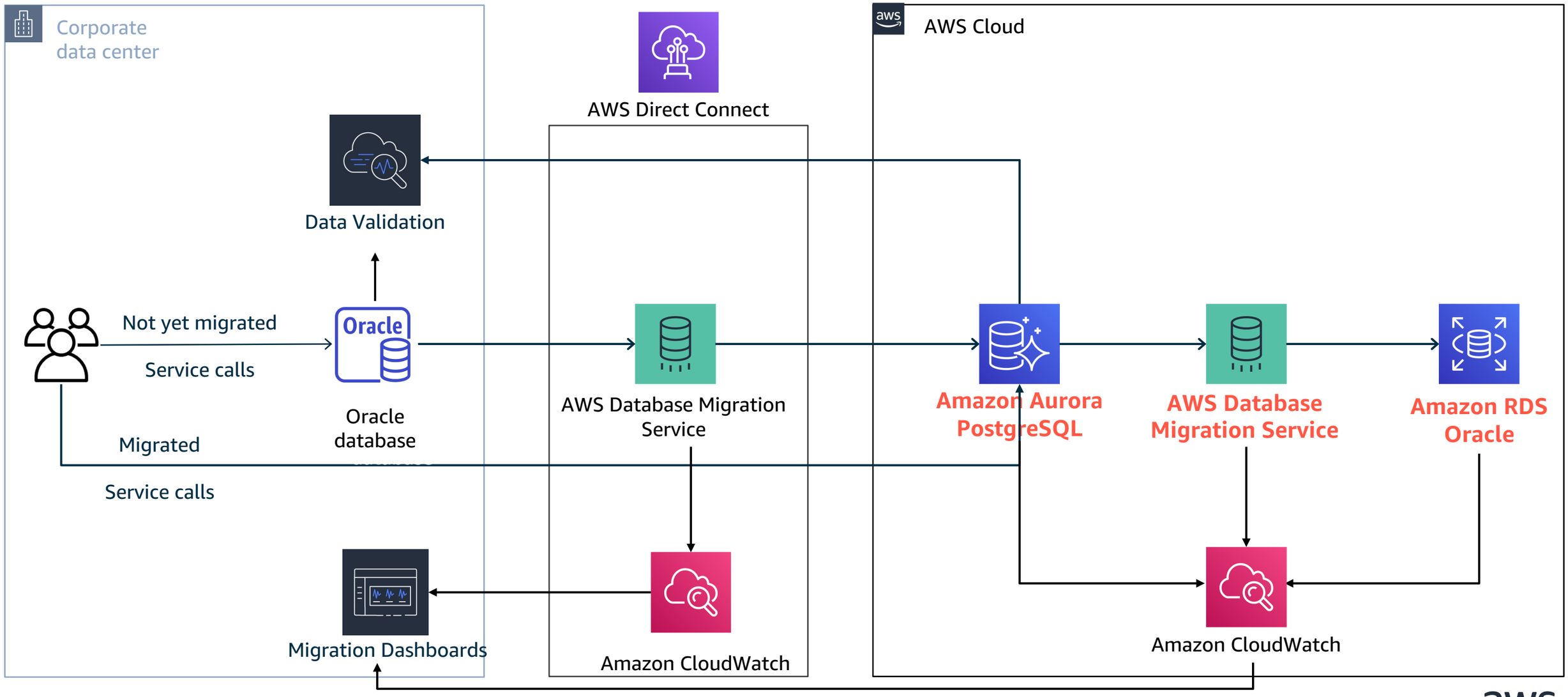
OracleからAmazon RDS/Aurora PostgreSQLへの移行



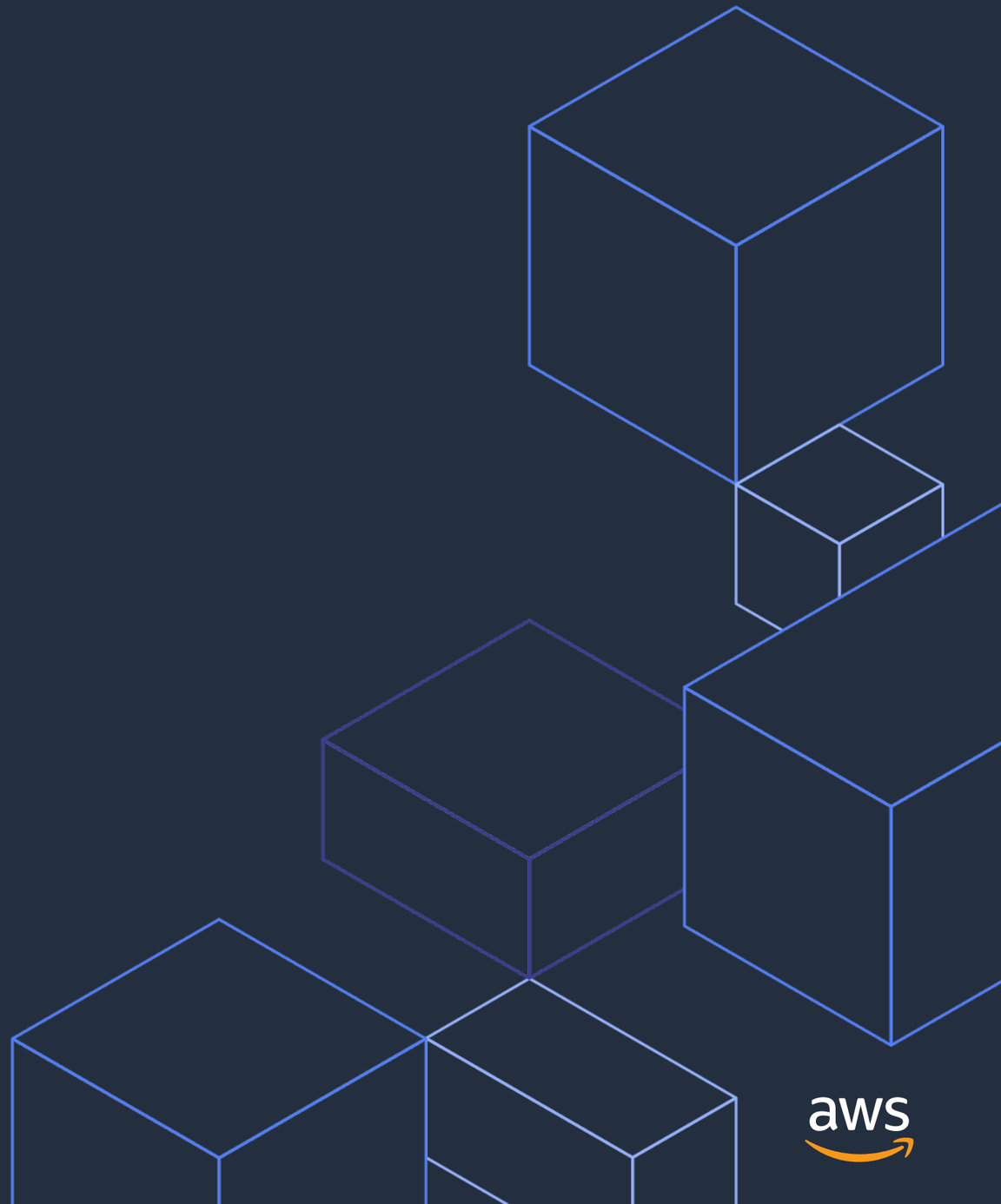
OracleからAmazon RDS/Aurora PostgreSQLへの移行



OracleからAmazon RDS/Aurora PostgreSQLへの移行

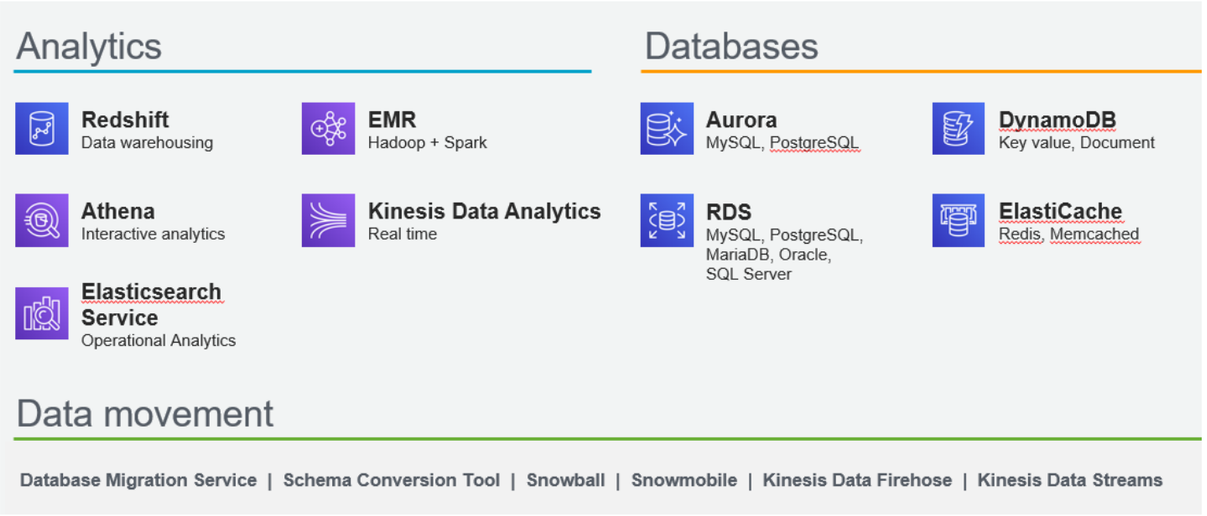


移行の効果



結果

- 約7,500 のデータベースをOracleから Amazon RDSもしくはAuroraに移行
- 303以上のビジネスクリティカルなサービスをDynamoDBに移行
- 運用管理コストを削減
- レイテンシーの向上とイノベーションを実現
- プライムデーの様なイベントの際のシステム拡張作業工数の削減



移行の効果

Purpose-built データベース により適材適所の選択が可能になった

万能のデータベースは存在しない

コスト削減

運用管理コストは
40%～90%の削減

パフォーマンス向上

処理量が2倍～4倍に増加し
レイテンシーは40%削減

スケーリング時の
作業工数削減

AWSのマネージドデータベース
を利用する事でピーク時の
スケーリング作業工数が1/10に

https://pages.awscloud.com/AWS-Innovators-Amazon.html

How Amazon is Achieving Database Freedom Using AWS

If a company like Amazon can move so many databases used by so many decentralized, globally distributed teams from Oracle to AWS, it's really within the reach of almost any enterprise.

Thomas Park, Senior Manager of Solution Architecture for Consumer Business Data Technologies, Amazon

Read More

Amazon.com is the world's largest online retailer. Amazon is guided by four principles: customer obsession rather than competitor focus, passion for invention, commitment to operational excellence, and long-term thinking. Customer reviews, 1-Click shopping, personalized recommendations, Prime, Fulfillment by Amazon, AWS, Kindle Direct Publishing, Kindle, Fire tablets, Fire TV, Amazon Echo, and Alexa are some of the products and services pioneered by Amazon.

amazon

Learn How Leading Cloud Innovators Build on AWS

Learn how leading cloud innovators are building on AWS to drive innovation and help shape the future of cloud services.

AWS provides these enterprises with a broad set of products and services they use as building blocks to run sophisticated and scalable applications. Running applications in the AWS Cloud can help them move faster, operate more securely, and save substantial costs; all while benefitting from the scale and performance of the cloud.

Read more »

aws

OracleからPostgreSQLへ移行する場合の技術的なポイント

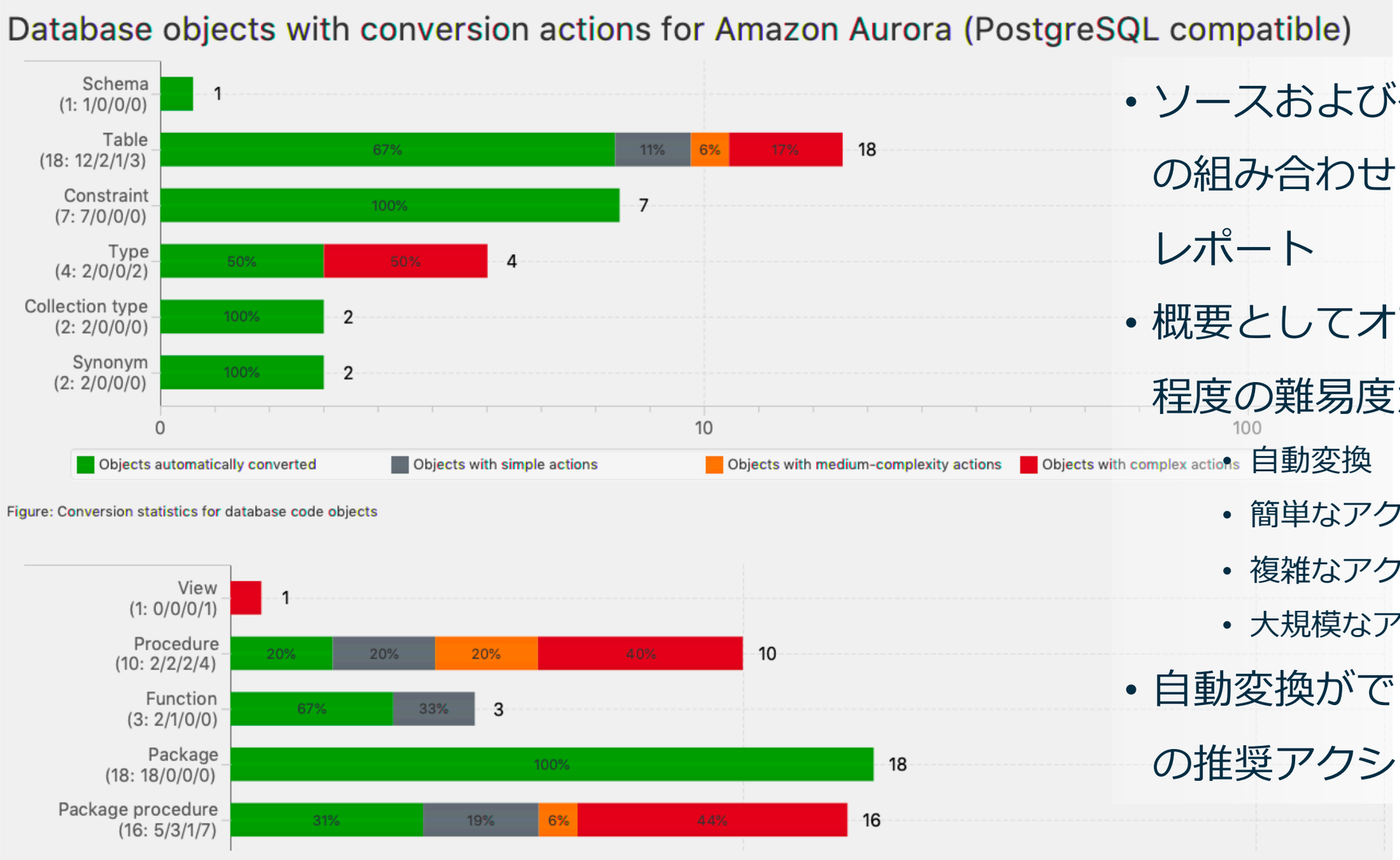
AWSのサービスを利用したデータベース移行の仕組みと流れ

Assessment	Schema conversion	Code conversion	Data migration	Validation
• スキーマ、SQL移行の難易度を調査	• スキーマの移行	• SQL、プロシージャーの移行	• データの移行	• データ移行の正常性をテスト

SCT
Schema Conversion Tool

DMS
Database Migration Service

AWS Schema Conversion Tool(SCT)の評価レポート



- ソースおよびターゲットデータベースの組み合わせに合わせたアセスメントレポート
- 概要としてオブジェクトの移行がどの程度の難易度か表示
  - 自動変換
    - 簡単なアクションが必要(1時間以内)
    - 複雑なアクションが必要(4時間以内)
    - 大規模なアクションが必要(4時間以上)
- 自動変換ができない場合は詳細レベルの推奨アクションを提示

SCTの評価レポートでOracleからPostgreSQLへの移行時によく指摘されるポイント

- 5027: Unable to convert object since there is no package body source code provided
- 5028: Unable to convert unsupported datatypes
- 5029: Unable to convert simple usage of unsupported datatype objects
- 5030: Unable to convert complex usage of unsupported datatype objects
- 5075: PostgreSQL doesn’t support VIEW with the READ ONLY option
- 5103: Unable to convert hints
- 5225: PostgreSQL doesn't support TYPE ... IS REF CURSOR declaration
- 5126: PostgreSQL doesn’t support TYPE ... IS REF CURSOR usage
- 5243: Converted trigger was owned by another schema. PostgreSQL always create trigger under the table’s schema
- 5271: The Oracle GREATEST function and PostgreSQL GREATEST function might give different results
- 5272: The Oracle LEAST function and PostgreSQL LEAST function might give different results
- 5334: Unable to convert statements with dynamic SQL statement
- 5340: Unable to convert functions
- 5350: Unable automatically convert statements that explicitly apply or cancel a transaction
- 5550: PostgreSQL doesn't have a data type ROWID
- 5554: PostgreSQL doesn't support virtual columns
- 5561: PostgreSQL doesn’t support this pre-defined exception
- 5571: PL/SQL pragma is unsupported
- 5572: PostgreSQL doesn’t support object type methods
- 5576: Unparsed SQL
- 5578: Unable to automatically transform the SELECT statement
- 5580: The exception block was emptied after conversion
- 5581: PostgreSQL doesn’t support index-organized table
- 5584: Converted functions depends on the time zone settings
- 5586: Automatic conversion for queries with NOCYCLE clause not supported 5591: Unable to convert system object
- 5598: PostgreSQL doesn't support a ROWID
- 5600: PostgreSQL doesn’t support a reference to SQLERRM with any specific parameter value
- 5602: PostgreSQL error code type is not a number and is incompatible with the number type variables
- 5605: PostgreSQL doesn't support use of dblinks
- 5608: Unable to convert the UPDATE statement with multiple-column subquery in SET clause
- 5610: This statement references to a synonym
- 5617: Converted code execution results might differ from the expected ones
- 5621: Converted code might be incorrect
- 5624: Converted code might be incorrect because of the bind variable names
- 5627: Converted code might be incorrect because of the use of UDF
- 5628: Converted code might be incorrect because of the dynamic SQL
- 5633: Conversion of dynamic SQL statement is not supported for DBMS_SQL.PARSE procedure
- 5643: BULK COLLECT INTO multilevel collection is not supported
- 5644: Unable automatically convert assign operation of array or global nested table, because of nested record
- 5647: FETCH BULK COLLECT INTO is not supported
- 5651: Current implementation of TABLE() does not support pipelined functions
- 5652: NULL columns not supported for partitioning
- 5655: Update of the partitioned table, its partition or subpartition may lead to errors
- 5656: TIMESTAMP. Partition contents could vary from Oracle
- 5661: INTERVAL. Data distribution could vary from Oracle. You may need to create partitions or subpartitions manually.
- …

オブジェクトの互換性

オブジェクト名の仕様の違い

- オブジェクト名の大文字、小文字の違い

  Oracleのオブジェクト名(ex. スキーマ名、テーブル名、カラム名など)のデフォルトは大文字ですが、PostgreSQLは小文字となっています。
  移行時にオブジェクト名に”(ダブルクォート)をつけて移行するとPostgreSQL側でも大文字を強制することになり、PostgreSQLで問題となります。

  例) “TABLE_A”の場合

  select * from table_AはOracleでは問題ありませんが、PostgreSQLではエラーになります

  SCT(Schema Conversion Tool)ではデフォルトでこの違いを吸収して変換されます。

  * SCTではOracleで明示的に大文字で作成されたオブジェクトも小文字に変換されるので注意が必要です

NULL columns not supported for partitioning

NULL columns not supported for partitioning (1/2)

原因
Oracleのパーティションテーブルではnot NULL制約がないカラムをパーティションキーにできますが、PostgreSQLではパーティションキーとなるカラムにnot NULL制約が必要なため、not NULL制約のないカラムをパーティションキーにするとこのエラーが報告されます。PostgreSQL 11以降ではDEFAULTパーティションに格納されます。

対処
SCTから該当箇所を確認し、手動でSQL文を変更する必要があります。

NULL columns not supported for partitioning (2/2)

```
-- Oracle SQL Snippet
create table dt_part_table (dt date, c1 number)
partition by range(dt)
(partition p1_2019_q1 values less than
                    (to_date('20190401','yyyymmdd’)),
 partition p2_2019_q2 values less than
                    (to_date('20190701','yyyymmdd')));
```

パーティションのサポート状況

- パーティション

  PostgreSQL 10系から宣言的パーティションをサポートしていますが、PostgreSQL 11系未満では、HASHパーティションをサポートしていません。そのため、OracleのHASHパーティションからの移行先をどのタイプのパーティションにするか検討する必要あります。(以下に記載していないパーティションについても同様）HASHパーティションの主な目的は、I/O分散とパーティション間のデータ量均一化であるため、他のパーティションタイプに移行する場合には、同様の観点での検討が必要となります。

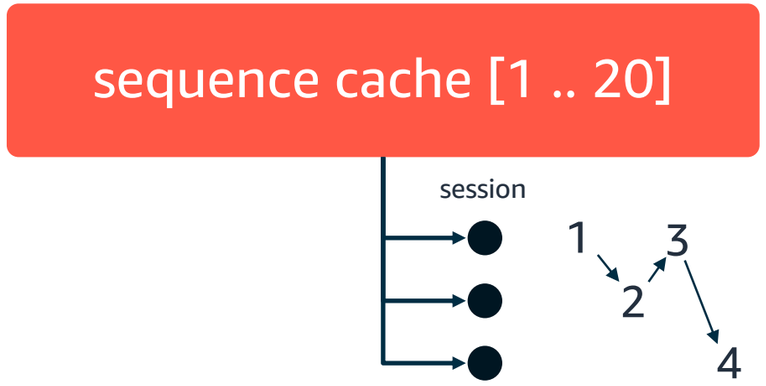
パーティション タイプ	Oracle Database	PostgreSQL (10.x互換)	PostgreSQL (11.x互換)	AWS SCT による変換
RANGE	○	○（宣言的パーティション）	○（宣言的パーティション）	○
LIST	○	○（宣言的パーティション）	○（宣言的パーティション）	○
HASH	○	N/A	○（宣言的パーティション）	N/A(10.x) ○(11.x)

その他オブジェクトの仕様の違い/サポートの有無

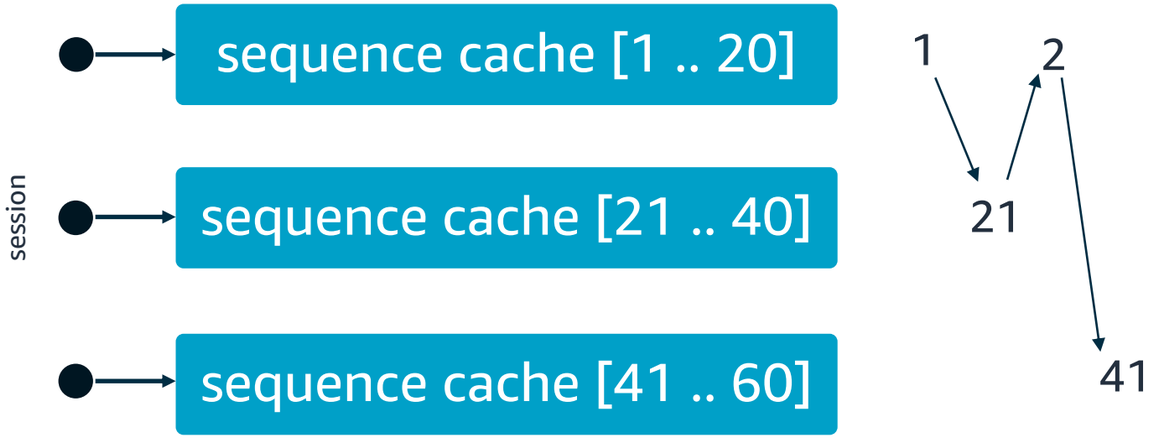
- シーケンス

  シーケンスのキャッシュの設定がOracleとPostgreSQLとで意味が異なる(Oracleはインスタンス単位、PostgreSQLはセッション単位)ため、シーケンスの順番を意識するプログラムでは注意が必要です。また、DMSを利用してデータ移行を行う場合、シーケンスの値を同期することができません。ですのでデータ移行後のシステム切り替え前にシーケンスの値を移行元より大きい値に設定し直す必要があります。

Oracleの場合



PostgreSQLの場合 (デフォルト値は1)



その他オブジェクトの仕様の違い/サポートの有無

- マテリアライズドビュー

  PostgreSQLにもOracleと同等のマテリアライズドビューが存在するもののOracleの高速リフレッシュ(FAST REFRESH)に対応する機能がありません。
  そのためMaterialized View Logを利用した差分での高速リフレッシュが必要な場合は、完全リフレッシュでパフォーマンス的に問題ないか、アプリケーションでの差分適用の仕組みを構築できるかなどを検討する必要があります。

  また、PostgreSQLのマテリアライズドビューは読み取り専用のみサポートしているので、更新可能(マルチマスター)マテリアライズビューはサポートしていません。

SQL、関数、データ型の互換性

The Oracle GREATEST/LEAST function and PostgreSQL GREATEST/LEAST function might give different results

aws

The Oracle GREATEST/LEAST function and PostgreSQL GREATEST/LEAST function might give different results

原因

OracleのGREATEST/LEAST関数とPostgreSQLのGREATEST/LEAST関数の仕様の違いにより返ってくる結果が異なる可能性あります。

対処

GREATEST関数の引数にNULLがある場合、 PostgreSQLはNULLを無視しますが、その他のデータベース(Oracleを含む多くのデータベース)はNULLがあればNULLを返すので、その仕様の違いを考慮する必要があります。

構文としては問題ないが結果が異なるパターン(1/2)

- NULLと空文字
  - Oracleの''(空文字表現)はNULLと同義だが、PostgreSQLではNULLと''(空文字)は別の意味

    INSERT INTO T VALUES ('');

    ⇒ Oracleの''(空文字)は明示的にNULLと書き換えが必要

  - 文字列連結(||)でNULLを連結する際にOracleは''(空文字)として連結するが、PostgreSQLはNULLを厳密に扱うので連結後の文字列はNULLとなる(機械的に''をNULLに置き換えると問題になる場合がある)

    SELECT 'TEST' || NULL FROM DUAL;

    ⇒ Oracleは'TEST'が返ってくるが、PostgreSQLはNULLが返る

    ⇒ ||の代わりにCONCAT_WS()を利用する

構文としては問題ないが結果が異なるパターン(2/2)

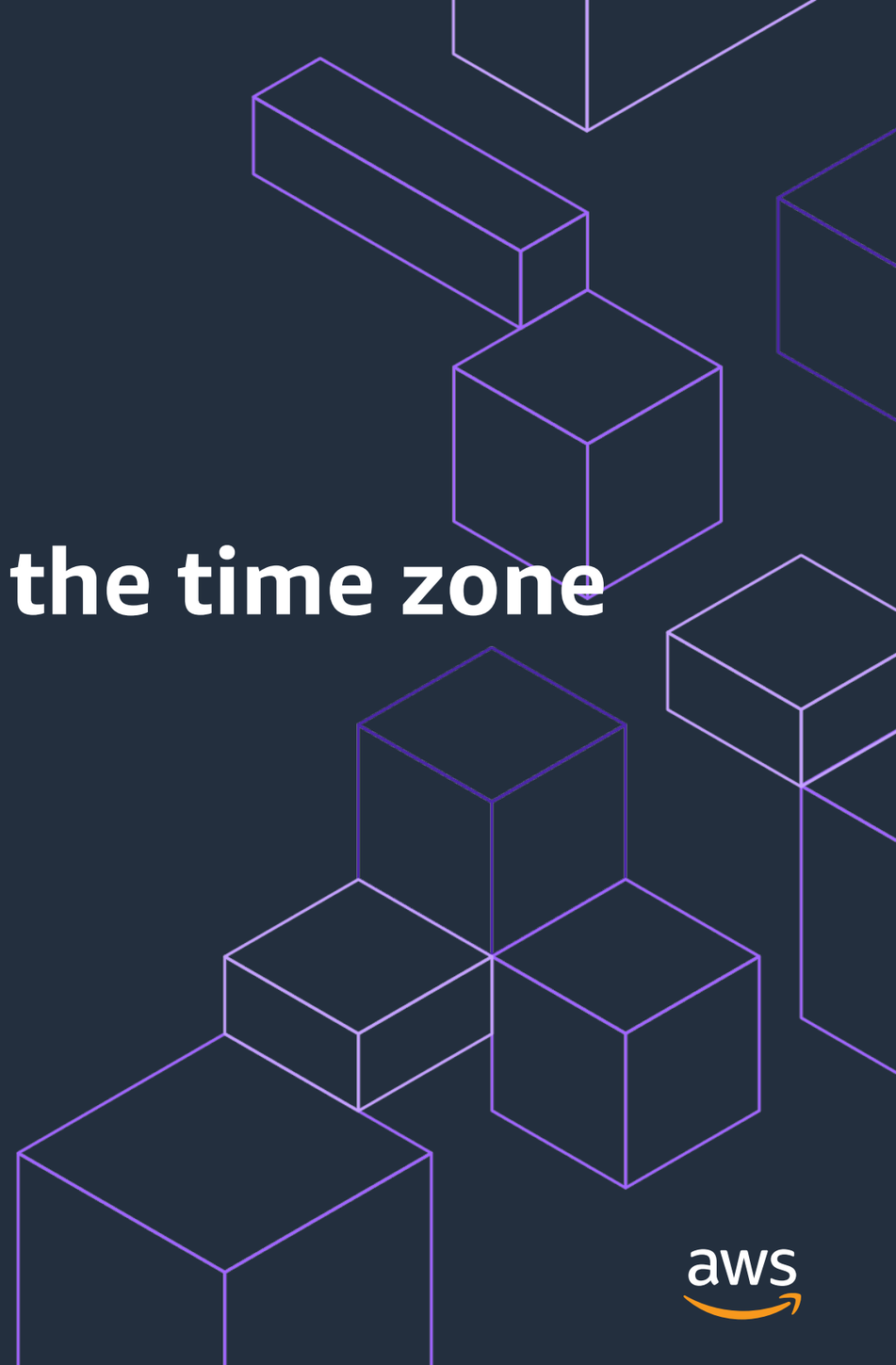
- CHAR型の末尾にパディングされた空白の扱いの違い

```
CREATE TABLE t1 (v char(5));
INSERT INTO t1 VALUES ('aws');

1: SELECT v || 'cloud' FROM t1;
2: SELECT CONCAT(v, 'cloud') FROM v1;
3: SELECT LENGTH(v) FROM t1;
```

- Oracleの場合
  - 1: aws cloud
  - 2: aws cloud
  - 3: 5
- PostgreSQLの場合
  - 1: awscloud
  - 2: aws cloud
  - 3: 3

Converted functions depends on the time zone settings



Converted functions depends on the time zone settings

原因
タイムゾーンに依存するデータ型に関連する関数などの変換を行う場合にこのエラーが報告されます。

対処
通常、手動の対応は必要ない場合が多いですが、SCTから該当箇所を確認し、タイムゾーンに依存する処理がないか確認が必要です。一般的にSYSDATEを使用している部分で確認が必要です。

日付/時刻型での注意点(1/2)

- 日付/時刻

  OracleのDATE型はPostgreSQLのTIMESTAMP without timezone型に変換が必要です。また、Oracleでの日付型の演算はPostgreSQLでは書き換える必要があります。

  OracleではDATE型同士の直接的な演算、DATE型、TIMESTAMP型への数値リテラルの演算がサポートされていますが、PostgreSQLではDATE型ではなくTIMESTAMP型での演算が必要です。また、TIMESTAMP型への数値リテラルの演算がサポートされていないので期間リテラルへの変換が必要です。

  select date_col + **1** from sample;

  ↓

  select date_col + **(1::NUMERIC || ‘ days’)::INTERVAL** from sample;

日付/時刻型での注意点(2/2)

演算パターン (データ型　演算子 データ型 => 結果のデータ型)		AWS SCTによる変換
Oracle Database	PostgreSQL	
DATE型 - DATE型 => NUMBER型	(EXTRACT (EPOCH FROM TIMESTAMP型 – TIMESTAMP型) / 86400)::NUMERIC => NUMERIC型	○
(DATE型 – DATE型) + 数値リテラル => NUMBER型	(EXTRACT (EPOCH FROM TIMESTAMP型 – TIMESTAMP型) / 86400)::NUMERIC + 数値リテラル => NUMERIC型	○
DATE型 + | – 数値リテラル => DATE型	TIMESTAMP型 + | – 期間リテラル => TIMESTAMP型	○
DATE型 + | – 期間リテラル => DATE型	TIMESTAMP型 + | – 期間リテラル => TIMESTAMP型	○
TIMESTAMP型 – TIMESTAMP型 => INTERVAL DAY TO SECOND型	TIMESTAMP型 – TIMESTAMP型 => INTERVAL型	手動変換/SCT
(TIMESTAMP型 – TIMESTAMP型) + 期間リテラル => INTERVAL DAY TO SECOND型	(TIMESTAMP型 – TIMESTAMP型) + 期間リテラル => INTERVAL型	手動変換/SCT
TIMESTAMP型 + | – 数値リテラル => DATE型	TIMESTAMP型 + | – 期間リテラル => TIMESTAMP型	手動変換/SCT
TIMESTAMP型 + | – 期間リテラル => TIMESTAMP型	TIMESTAMP型 + | – 期間リテラル => TIMESTAMP型	手動変換/SCT

PostgreSQL doesn't support a ROWID

PostgreSQL doesn't support a ROWID

原因
ROWID型、ROWID擬似カラムはPostgreSQLには存在しないのでそれらを定義、参照する場合このエラーが報告されます。

対処
PostgreSQL10以降で、SCTのROWIDをエミュレートする設定をONにした場合、各テーブルにROWIDカラムがBIGINT型で追加されます。またROWIDカラムはGENERATED ALWAYS AS ITENTITYとして自動採番の数値が登録されます。この仕組みを使う場合、ROWID型はBIGINT型にすることで対処が可能になります。

https://docs.aws.amazon.com/ja_jp/SchemaConversionTool/latest/userguide/CHAP_Source.Oracle.ToPostgreSQL.html

Unable to convert the UPDATE statement with multiple-column subquery in SET clause

aws

Unable to convert the UPDATE statement with multiple-column subquery in SET clause (1/2)

原因

UPDATE文のSET句に複数カラムを指定し、変更後の値をSELECT文のサブクエリーとして指定している場合にこのエラーが報告されます。

対処

SCTから該当箇所を確認し、手動でSQL文を変更する必要があります。

Unable to convert the UPDATE statement with multiple-column subquery in SET clause (2/2)

```
-- Oracle SQL Snippet
update t1
set (c1,c2) = (select max(c1),min(c1) from t2) ;

↓

-- PostgreSQL SQL Snippet
update t1
set c1=q.max_c1, c2=min_c2
from (select max(c1) max_c1, min(c1) min_c1 from t2) q ;
```

ストアドプロシージャの互換性

Unable to convert statements with dynamic SQL statement

Unable to convert statements with dynamic SQL statement (1/2)

原因

PL/SQL内で文字列でSQL文を組み立てるなどして動的なSQL文を実行する場合このエラーが報告されます。

対処

SCTから該当箇所を確認し、手動でロジックの変換を行う必要があります。通常Oracleの動的SQL文の実行はPostgreSQLのEXECUTE文で代替可能です。しかし生成される動的なSQL文が不定なので、それらがPostgreSQL上で問題なく動作するか確認する必要があります。

Unable to convert statements with dynamic SQL statement (2/2)

```
-- PL/SQL code snippet
…
begin
     execute immediate 'select c1 from t where c1='||var1 into i;

    …

-- PL/pgSQL code snippet
…
begin
    execute 'select c1 from t where c1='||var1 into i;
    …
```

PostgreSQL doesn‘t support this pre-defined exception

PostgreSQL doesn't support this pre-defined exception (1/2)

原因

PL/SQLで事前定義されている例外のうち自動変換できなかったものがある場合にこのエラーが報告されます。(ACCESS_INTO_NULL, COLLECTION_IS_NULL, NO_DATA_NEEDED, ROWTYPE_MISMATCH, SELF_IS_NULL, SUBSCRIPT_BEYOND_COUNT, SUBSCRIPT_OUTSIDE_LIMIT, SYS_INVALID_ROWID, USERENV_COMMITSCN_ERRORなど)

対処

Oracle独自の処理内容に沿った例外である場合もありPostgreSQLに移行する場合、既存(PL/SQL)で必要だった例外が不要になる場合もあるため前後の処理を確認してください。または、複雑な例外処理にならないよう処理全体を書き直す必要があります。

PostgreSQL doesn't support this pre-defined exception (2/2)

```
-- PL/SQL code snippet
create or replace function return number is
begin
    …
exception
    when access_into_null then
        …
    when zero_divide then
        …
    when others then
        …
end;
/
```

PostgreSQL doesn’t support a reference to SQLERRM with any specified parameter value

PostgreSQL doesn’t support a reference to SQLERRM with any specified parameter value (1/2)

原因

PL/SQLの中にSQLERRMを引数付きで使用している場合、このエラーが報告されます。

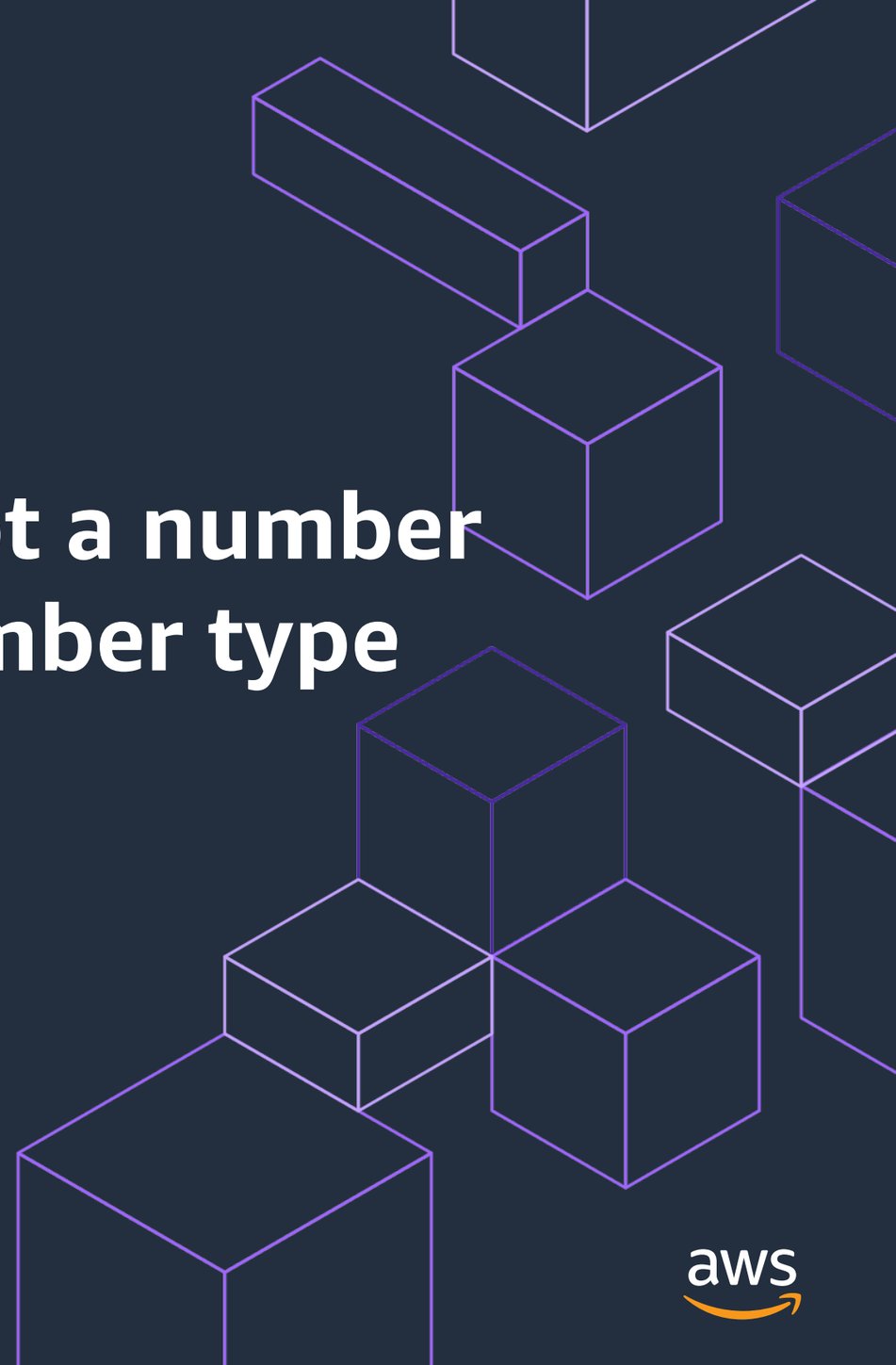
対処

PL/SQLの中で利用しているSQLERRM(SQLエラーコード)の書式が直近のエラーメッセージを出力する単純なSQLERRMに書き換えられないか確認してください。

PostgreSQL doesn’t support a reference to SQLERRM with any specified parameter value (2/2)

```
-- PL/SQL code snippet
create or replace procedure xxx
as
    …
begin
    …
exception
    when others then
        dbms_output.put_line(SQLERRM); <- これは問題なし
        dbms_output.put_line(SQLERRM(-1));
        dbms_output.put_line(SQLERRM(SQLCODE));
end;
/
```

PostgreSQL error code type is not a number and is incompatible with the number type variables



aws

PostgreSQL error code type is not a number and is incompatible with the number type variables (1/2)

原因

PL/SQLの中にSQLCODEを数値型として扱っている場合このエラーが発生します。

対処

PL/SQLの中で利用しているSQLCODEは、PostgreSQLでは存在しないためSQLSTATEに書き換えができないか検討していください。また、SQLCODEをSQLSTATEに変更する場合、数値型から文字列型に変更する必要があるため注意が必要です。

PostgreSQL error code type is not a number and is incompatible with the number type variables (2/2)

```
-- PL/SQL code snippet
is
    error_code number;
exception
    when others then
        error_code := SQLCODE;
        …

-- PL/pgSQL code snippet
is
    error_code varchar(100);
exception
    when others then
        error_code := SQLSTATE;
```

データベース移行の役にたつリソース

https://aws.amazon.com/jp/dms/resources/

Q&A

アンケートにご協力ください。

https://bit.ly/2S6VTUY

Thank you!