PostgreSQL Conference Japan 2021

NTT DATA
Trusted Global Innovator

【B4】OracleからPostgreSQLへのDB移行の実際

2021年11月12日

株式会社NTTデータ　薄 浩之

自己紹介

薄　浩之（うすき ひろゆき）

- 株式会社NTTデータ 勤務
- 2019年より、DB移行案件への技術支援を中心に活動中

  関わった案件：　70案件以上

概略

Oracle→PostgreSQLにDB移行する際の課題、解決方法

1. なぜPostgreSQLに移行するのか
2. 移行ツールで移行コストは下がるのか
3. 移行しやすいシステム、移行しにくいシステム
4. 技術的な課題とその対処
   - 数値演算結果の差異
   - 一時ファイル用領域をどうすべきか
   - ~~思わぬOut Of Memory~~
   - NULL/空文字問題
   - 拡張機能の選択

なぜPostgreSQLに移行するのか

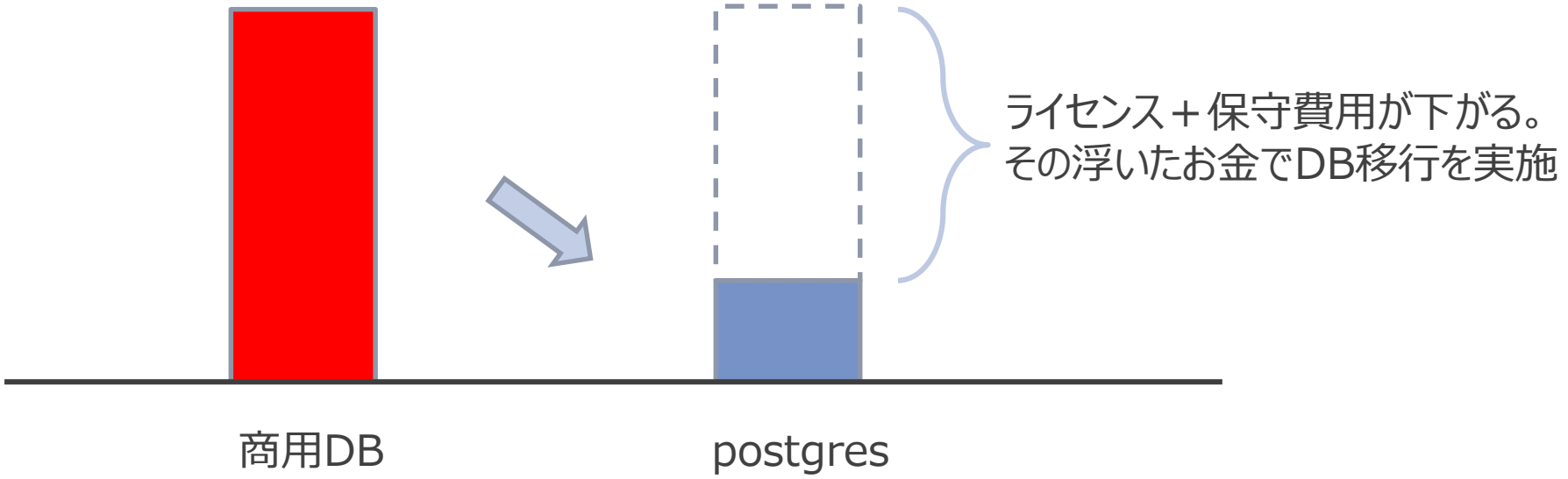
なぜPostgreSQLに移行するのか

結局、「コスト削減」

ほとんどのお客様は、OSS-DBに移行することで単純にシステム維持コストが下がることを期待している。つまり、、

DB移行でコスト負担が増えるなら、移行などしない

DB移行是非の判断



商用DBからOSS-DBに乗り換えることで、新規ライセンスの購入コスト、保守コストが節約できるが、その**「節約分のコスト内でDB移行を実現」**できなければＧＯは出ない

【判断基準】

商用DBをあと５年使い続けるコスト ＞ OSS-DBに乗り換え＋５年運用するコスト

DB移行是非の判断

【判断基準】

商用DBをあと５年使い続けるコスト ＞ OSS-DBに乗り換え＋５年運用するコスト

しかし、これはなかなか難しい・・・

まず、この試算を行い、DB移行にかけられるコスト枠を見極めることが必要

商用DB継続

- ハード増強等をしなければ新規ライセンス購入は不要
- 商用DB継続の場合もバージョンアップ対応の工数等はかかる
- クラウド移行する場合のライセンス料変化

OSS-DB乗り換え

- OSS-DBでも保守サポート契約は必要
- OSS-DBのEOLにより運用中のバージョンアップは発生する
- プラス、DB移行のコスト

DB移行是非の判断

【判断基準】

商用DBをあと５年使い続けるコスト ＞ OSS-DBに乗り換え＋５年運用するコスト

しかし、これはなかなか難しい・・・

まず、この試算を行い、DB移行にかけられるコスト枠を見極めることが必要

商用DB継続

- ハード増強等をしなければ新規ライセンス購入は不要
- 商用DB継続の場合もバージョンアップ対応の工数等はかかる
- クラウド移行する場合のライセンス料変化

OSS-DB乗り換え

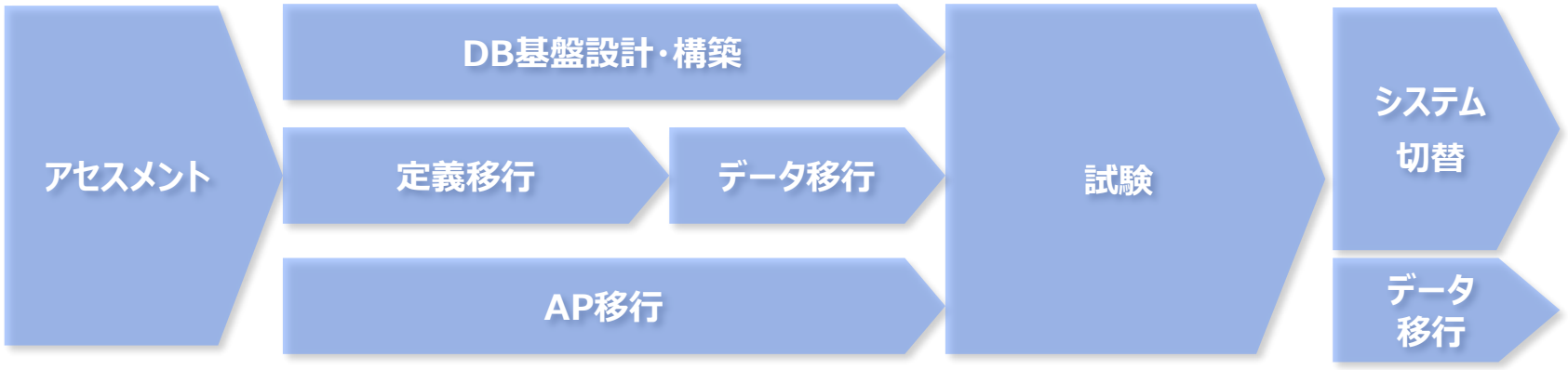
- OSS-DBでも保守サポート契約は必要
- OSS-DBのEOLにより運用中のバージョンアップは発生する
- **プラス、DB移行のコスト**

DB移行コストの見積りは難しいため、他を見積り、まずはDB移行コストの枠を明らかにすべし

PostgreSQLへの移行コストは何で決まる？

DB移行案件のタスクとは

このそれぞれのタスクの工数によりコストが決まる

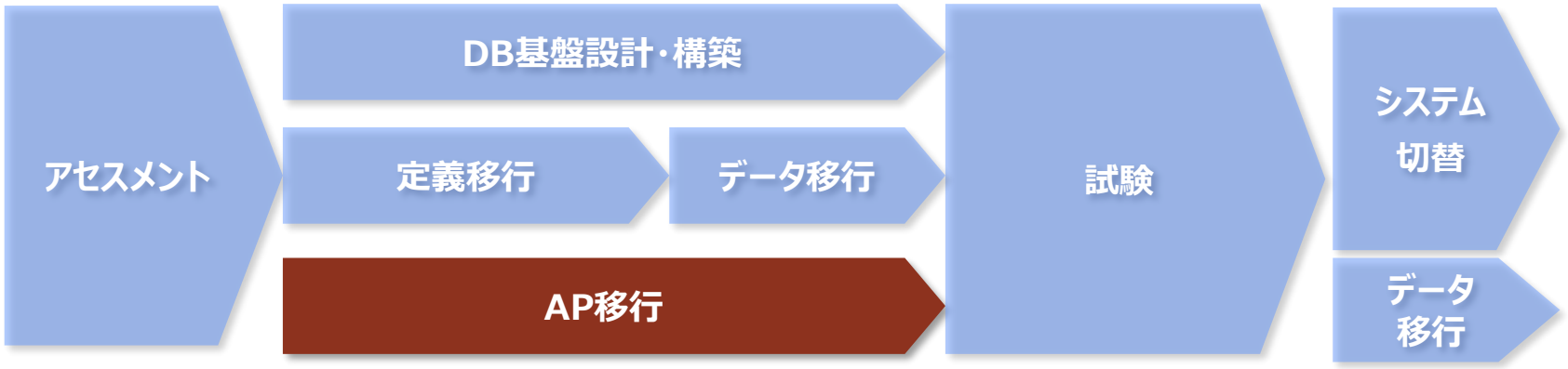


NTT DATA

PostgreSQLへの移行コストは何で決まる？

DB移行案件のタスクとは

このそれぞれのタスクの工数によりコストが決まる



システムにより大きく変動しやすいのは「AP移行」のコスト

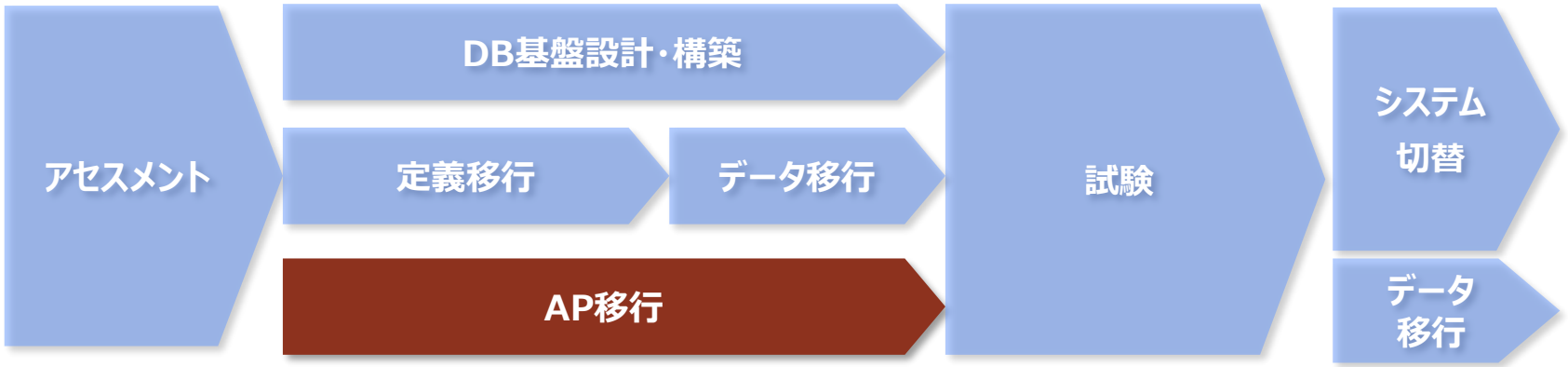
- AP規模
- 使用している言語
- PostgreSQLに移行困難な機能の使用状況
- AP移行ノウハウを保有しているか

NTT DATA

PostgreSQLへの移行コストは何で決まる？

DB移行案件のタスクとは

このそれぞれのタスクの工数によりコストが決まる



システムにより大きく変動しやすいのは「AP移行」のコスト

- AP規模
- 使用している言語
- PostgreSQLに移行困難な機能の使用状況
- AP・SQL移行ノウハウを保有しているか

コストを下げるには？

効率的なAP移行作業

言語別の移行ノウハウ

代替機能のノウハウ

AP・SQL移行ノウハウ

DB移行を考えるその他の動機

コスト以外には下記のような理由が挙がる

1. ベンダロックインの回避
2. システムアジリティの向上（クラウド移行とセットで考えることが多い）
3. よりシステム用途に合った適切なDBの選択

DB移行を考えるその他の動機

コスト以外には下記のような理由が挙がる

1. ベンダロックインの回避
2. システムアジリティの向上（クラウド移行とセットで考えることが多い）
3. よりシステム用途に合った適切なDBの選択

その点、PostgreSQLは移行先のOSS-DBとして非常に良い候補

開発コミュニティの独立性

活発な開発が続いている

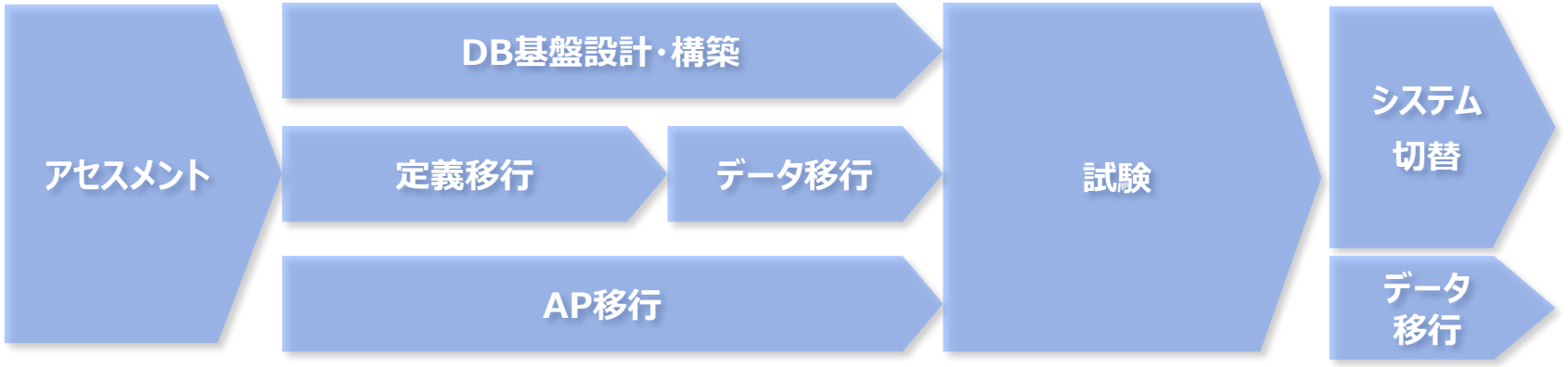
大手クラウドサービスでマネジメントサービスとして提供

様々な拡張機能、カスタマイズ版PostgreSQLで特殊用途にも対応

移行ツールで
移行コストは下がるのか

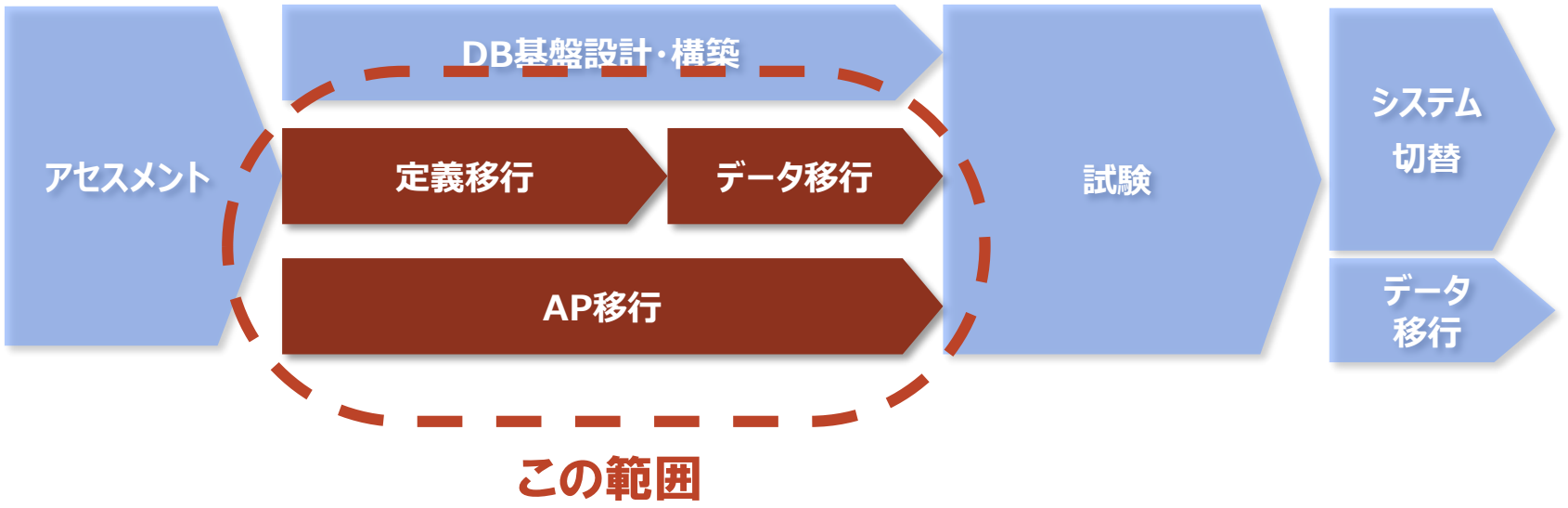
移行ツールで移行コストを削減？

そもそも移行ツールが“効く”タスクは？



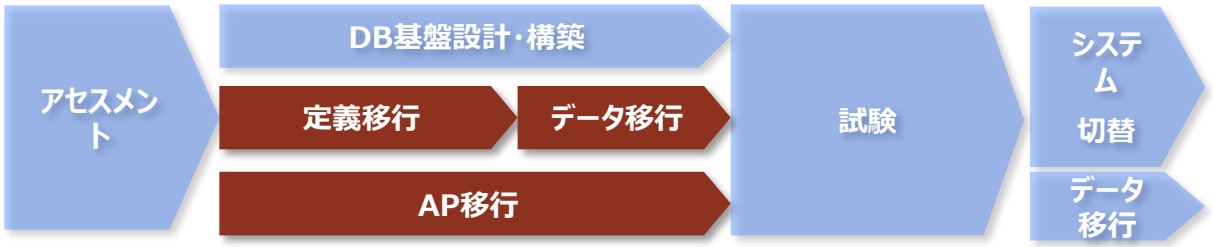
移行ツールで移行コストを削減？

そもそも移行ツールが“効く”タスクは？



移行ツールで移行コストを削減？

主な移行ツール



ツール		Ora2pg	MnMTK	DMS(SCT)	EDB Migration Toolkit
開発元		OSS	Ispirer	AWS	EnterpriseDB
定義移行		●	●	●	●
データ移行		●	●	●	●
AP移行	PL/SQL	●	●	●	●
	java内SQL埋め込み			●	
	C内SQL埋め込み			●	
	Pro*C				
備考				AWSへの移行時のみ使用可	EDBへの移行時のみ使用可

移行ツールで移行コストを削減？

移行ツールに過度な期待をしてはいけない

- **AP資産が移行ツールで移行可能かどうか**

  まずはそもそもツールが移行対象としている言語であること。さらに、SQLの記述方法や使用しているDB機能によって変換精度が変わる。**ほとんど変換できない場合もある**ため、案件計画時に試すのが必須。

- **移行ツールが生成したコードの試験**

  移行ツールは書き換えの理由を説明しない。書き換えが正しいことを確認するブラックボックス試験が必要であり、**移行前DBと結果が同じになることを試験する**ことになる。

- **移行ツールが生成したコードのメンテナンス**

  システム運用中の機能追加など、コードに修正が入る場合がある。その際、**自動生成されたコードのメンテナンス性は決して高くない**ことを理解する。

移行しやすいシステム
移行しにくいシステム

（素の）PostgreSQLに移行しやすいシステム

【コストが見合う】

DBサーバスペックが高い（コア数が多い）

AP規模が小さい

【PostgreSQLの特性に向いている】

OLTP系である

システムにDBが一つだけ

（素の）PostgreSQLに移行しやすいシステム

【コストが見合う】

DBサーバスペックが高い（コア数が多い）

AP規模が小さい

【PostgreSQLの特性に向いている】

OLTP系である

システムにDBが一つだけ

DBサーバスペックが高い

（コア数が多い）

||

維持コストが高い

すなわち、OSS-DB移行によるコストメリットが生じやすい

（素の）PostgreSQLに移行しやすいシステム

【コストが見合う】

DBサーバスペックが高い（コア数が多い）

AP規模が小さい

【PostgreSQLの特性に向いている】

OLTP系である

システムにDBが一つだけ

AP規模が小さい

||

AP移行コストが抑えられる

これも、OSS-DB移行によるコストメリットが生じやすい

（素の）PostgreSQLに移行しやすいシステム

【コストが見合う】

DBサーバスペックが高い（コア数が多い）

AP規模が小さい

【PostgreSQLの特性に向いている】

OLTP系である

システムにDBが一つだけ

基本的なSQLについてはOracleと同等程度のレベルであり、OLTP系システムでは移行後の性能担保が比較的容易。

一方、大量データ処理が得意ではなく、DWHなど分析系システムの移行には工夫が必要。

・大量データロード

・大量データ集計

✓ PG_StromやCitusなどの拡張機能を検討

（素の）PostgreSQLに移行しやすいシステム

【コストが見合う】

DBサーバスペックが高い（コア数が多い）

AP規模が小さい

【PostgreSQLの特性に向いている】

OLTP系である

システムにDBが一つだけ

Oracleに比べDBリンク機能が使いづらく、分散DB構成のシステムのAP移行は大きな変更が必要になることが多い。

- postgreSQLのDBリンク機能はdblink、dblink_plus、postgres_fdwといったもの

例えば、「複数のOracle DBがDBリンクで強連携して稼働しているシステムについて、そのひとつをPostgreSQLに移行する」というのは移行が困難。

技術的な課題と
その対処

※ 特にOracle → PostgreSQLの移行における

DB移行のノウハウ

DB移行の各種ノウハウは、大量の公開リソースが存在。

- PGECons成果物
  https://www.pgecons.org/works_index/
  「データベース移行」（WG2）の各種ドキュメント

- AWSのMigration Playbook
  https://aws.amazon.com/jp/dms/resources/?nc=sn&loc=5
  「Oracle から Amazon Aurora PostgreSQL に移行する」のリンクで
  「Oracle Database 19c To Amazon Aurora with PostgreSQL Compatibility (12.4)」が公開中

数値演算結果の差異

- 精度指定なしのnumber、numeric

oracle

number : 1

number : 3

select (num1 / num2) * num2
from ～

結果 ⇒ **1**

postgreSQL

numeric : 1

numeric : 3

select (num1 / num2) * num2
from ～

結果 ⇒ **0.99999999999999999999**

精度指定なしのnumber型は浮動小数点型。postgreSQLではfloatにキャストすることで、ほとんど一致させられる。

postgreSQL

```
select ( num1::float / num2::float ) * num2::float from ～
```

結果 ⇒ **1**

数値演算結果の差異

- **整数リテラル**

oracle

select 1 / 3 from dual;

結果 ⇒ **0.3333333333**

select (1 / 3) * 3 from dual;

結果 ⇒ **1**

postgreSQL

select 1 / 3 ;

結果 ⇒ **0**

select (1 / 3) * 3 ;

結果 ⇒ **0**

postgreSQLは整数リテラルの演算は結果も整数となるように計算される。これもfloatにキャストすることで、ほとんど一致させられる。

postgreSQL

select 1**::float** / 3**::float** ;

結果 ⇒ **0.333333333333333**

select (1**::float** / 3**::float**) * 3**::float** ;

結果 ⇒ **1**

数値演算結果の差異

- 精度指定ありのNUMBER／NUMERIC型を使用していれば、基本的に結果は合う。
- 結果がずれる場合は**とりあえずfloatにキャスト**してみる

差異が出るとはいえ、どちらも計算結果が間違っているというわけではない。

しかし、DB移行前後で結果が変わるというのは受け入れられないことが多い。

一時ファイル用領域をどうすべきか

- 一時ファイル

  PostgreSQLは次の場合に一時的な作業ファイルを作る。

  - work_memに収まらないソート（ディスクソート）
  - work_memに収まらないハッシュ操作
  - 一時テーブル（Temporary Table）利用時

  つまり、**Oracleの一時セグメントと同様**の役割。

  デフォルトの場所は **PGDATA/base/pgsql_tmp**

普通にPostgreSQLの基盤設計をすると、この一時ファイル用の領域はデフォルトのテーブル空間（業務データなどを格納）と同じストレージパーティションになる。つまり、、

うっかり巨大なソート発生時などに、業務データ用のストレージFULLを引き起こし、DBが更新できない事態に！

一時ファイル用領域をどうすべきか

うっかり巨大なソート発生時などに、業務データ用のストレージFULLを引き起こし、DBが更新できない事態

- 対処

  一時ファイル用のストレージパーティションを独立させる。

  - 一時ファイル用ストレージ容量＝全セッション合計の一時ファイル容量制限となる
  - 合計の一時ファイル容量オーバー時には、当該SQLだけがエラーとなる
    （そしてそのSQLが確保していた一時ファイル領域は解放される）
  - 業務データのストレージを圧迫しない

  つまり、**Oracleの一時セグメントと同様**の動作となる

※ 後工程でストレージの切り直しをするのはハードルが高くなるため、DB基盤設計時に考慮しておくことが大事。

※ ちなみにtemp_file_limit（セッションあたりの一時ファイル最大容量）の設定で対処しようとするのは悪手

思わぬOut Of Memory

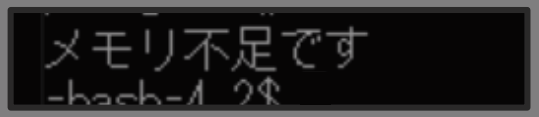
このケースですが、会場でご指摘があった通り、psqlがいったん結果セットを保持するために起きているOOMでした。
DB側で大量メモリを獲得してしまう再現ケースを用意できなかったため、このケースについて取り下げさせてください。

- 1万件程度のテーブルを作り、自分同士を直積結合させて

```
create table test (id integer);
insert into test select generate_series(1, 10000);    --10000件挿入

select * from test a, test b;    -- 結合条件なしで直積結合
```

結果



このテスト環境はメモリ4GB程度だったが、簡単にメモリを食いつぶし、Out Of Memory 。

※Oracleではこのようなことはほぼ起きない

SQL実行で消費するメモリ量上限は設定できない。
テスト済みの業務機能が発行するSQLでは、異常なメモリ消費が起こるリスクは限りなく小さい。

しかし、例えばBI系システムのユーザ自由検索などで想定しないSQLが組み立てられ実行されることがあり、これによるOOMリスクがある。

思わぬOut Of Memory

このケースですが、会場でご指摘があった通り、psqlがいったん結果セットを保持するために起きているOOMでした。
DB側で大量メモリを獲得してしまう再現パタンを用意できなかったため、このケースについて取り下げさせてください。

BI系システムのユーザ自由検索などで想定した
実行されることがあり、これによるOOMリスクが

- OOMは他セッションに大きな影響を及ぼす。DBが落ちる場合もあり、商用サービス提供しているDBでは起こしてはならない障害。
- 直積結合だけではなく、大量件数のテーブルを多数結合する場合なども危ない

十分テスト済みのSQLしか実行されないのであれば問題ないが・・・

自由検索インターフェースをユーザに提供し、任意のテーブル結合が行われるシステムでは・・・

【対処】自由検索をレプリカ側で実行し、最悪落ちてもいいDBとする

※ なお、Aurora等マネジメントサービスのPostgreSQLでも起こります。
（メモリ使用量を直接観測できないため、よりやっかい）

NULL/空文字 問題

- **OracleはNULLと空文字を区別しない**

Oracle	PostgreSQL
‘’(空文字)は ・NULL ・長さ０の文字列 場合に応じて両方の意味で扱われる	‘’（空文字）は長さ０の文字列 NULLも格納可能で、‘’とは別物

	Oracle	PostgreSQL	
	‘’（= NULL）	‘’（空文字）	NULL
WHERE **str** IS NULL	TRUE	FALSE	TRUE
length(**str**)	NULL	0	NULL
str || ’abc’	‘abc’	‘abc’	NULL

定型的なうまい移行方法が確立されていない

NULL/空文字 問題

- 対処

まずはOracleのNULL（=空文字）を、NULLか空文字か、どちらで移行するかを決める。
そして、対処方針に合わせて、データ移行やAP改修を行う。

NULL/空文字 問題

Oracleの‘’（＝NULL）を･･･

NULLとして移行	‘’(空文字) として移行
● データ移行時にNULLとして移行する ● insert/updateしようとしている値が空文字だった場合、NULLに変換する。	● データ移行時に ’’ として移行する ● insert/updateしようとしている値がNULLだった場合、空文字に変換する ● カラム追加等のテーブル定義変更時には暗黙的にNULLが入ってしまうので、それも’’に直す

	結果の違い	対策例	結果の違い	対策例
str IS NULL	–		結果がFALSEになる	**str IS NULL OR str = ''**
str || ‘abc’	NULLとなる	**CONCAT(str,’abc’)**	–	–
結合	–	–	colA = colB の両方が空文字だった場合に結合できてしまう。	**colA = colB and colA != '' and colB != ''**
文字列操作関数	trimやlength等の関数の結果がNULLになる。	**TRIM(COALESCE(col,‘’)) LENGTH(COALESCE(col,''))**	–	–
javaで値を取得する	string型の内容がNULLとなる	NULLを受け入れるようにAP修正。	–	–

拡張機能の選択

- 拡張機能

  PostgreSQLと言えば拡張機能。

  ワールドワイドで魅力的な拡張機能の開発が盛んで、開発中システムで大いに役立ちそうなものも見つかるはず。

しかし、マイナーな拡張機能はそのメンテナンスが中止されることも多く、次回のPostgreSQLアップグレード時に新バージョン対応版が無いなど、詰んでしまう事態に

【対処】

基本的には本体機能！

拡張機能は、数年後もメンテナンスが継続していそうな、メジャーなものを採用する

さいごに

DXケーススタディの発信サイト「デジタルテクノロジーディレクター®」
https://ndigi.tech/

新着記事

- ✓ **デジタル時代におけるデータベース移行戦略**
  https://ndigi.tech/all_post/14038
- ✓ **データベース移行で「攻める」ビジネスを**
  https://ndigi.tech/all_post/12987
- ✓ **企業のDXをさまたげるレガシー資産**
  https://ndigi.tech/all_post/11967