イチから覚える
AWS Schema Conversion Tool

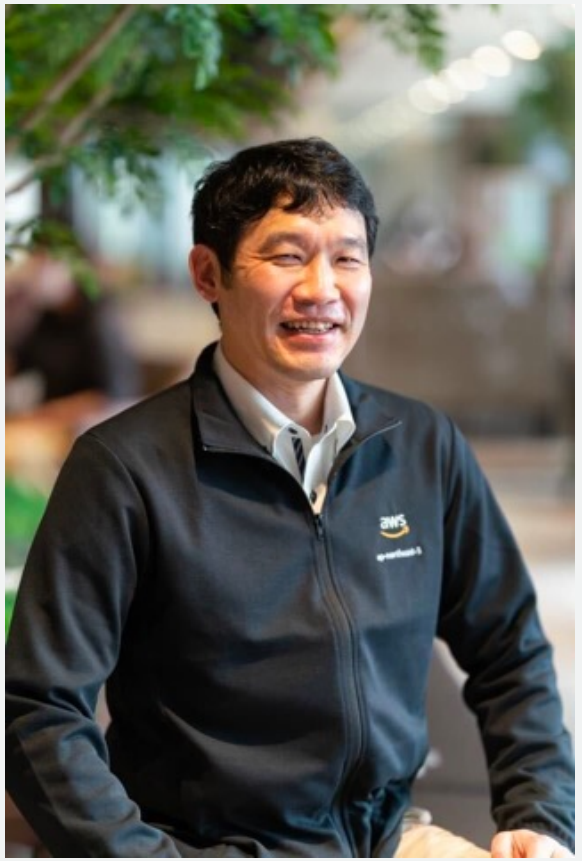
Yoshio Uchiyama

Senior Database Specialist Solutions Architect
Amazon Web Service Japan

自己紹介

内山　義夫

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



好きなAWSサービス

- Amazon RDS
- AWS Schema Conversion Tool
- AWS Database Migration Service

Agenda

- AWS Schema Conversion Tool とは
- AWS Schema Conversion Tool の基本的な使い方
- AWS Schema Conversion Tool ちょっぴりDive Deep
- まとめ

AWS Schema Conversion Tool とは

フルマネージド データベースサービスへの移行

データベース移行を支援するツールやサービス、各種プログラム

Assessment	Schema conversion	Code conversion	Data migration	Validation
• サイジング実施、スキーマ、SQL 移行の難易度を調査	• スキーマの移行	• SQL (DB、アプリ)、プロシージャ	• データの移行	• データ移行の正常性をテスト
• Schema Conversion Tool • Database Migration Service	• Schema Conversion Tool	• Schema Conversion Tool • Babelfish for Aurora PostgreSQL	• Database Migration Service	• Database Migration Service Data Validation
• Database Freedom Workshop(無償)で支援	• Database Migration Accelerator(有償)で支援	• AWS Professional Serviceやパートナー様のサービスで支援		

- データベース移行全般をAWS Professional Serviceやパートナー様のサービスで支援

データベース移行を支援するツールやサービス、各種プログラム

Assessment	Schema conversion	Code conversion	Data migration	Validation
• サイジング実施、スキーマ、SQL 移行の難易度を調査	• スキーマの移行	• SQL (DB、アプリ)、プロシージャ	• データの移行	• データ移行の正常性をテスト
• Schema Conversion Tool • Database Migration Service	• Schema Conversion Tool	• Schema Conversion Tool • Babelfish for Aurora PostgreSQL	• Database Migration Service	• Database Migration Service Data Validation

- Database Freedom Workshop(無償)で支援
- Database Migration Accelerator(有償)で支援
- AWS Professional Serviceやパートナー様のサービスで支援
- データベース移行全般をAWS Professional Serviceやパートナー様のサービスで支援

AWS Schema Conversion Tool (SCT)

Modernize & Migrate

データベースレイヤーの**モダナイズ**

スキーマ、ビュー、プロシージャ、ファンクションの大部分を**自動的に変換**

Oracle
SQL Server
IBM Db2
PostgreSQL
MySQL
Sybase
Azure SQL Database

→

Amazon Aurora
PostgreSQL
MySQL
MariaDB
Oracle
SQL Server

モダナイズと共にデータウェアハウスを Amazon Redshift へ**移行**

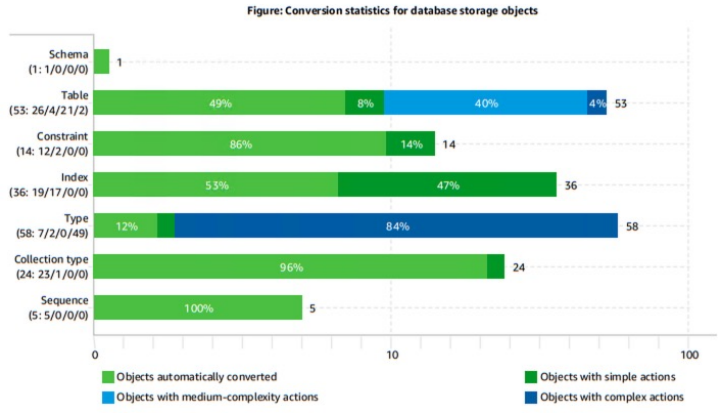
(* Snowball 連携も可能)

Greenplum
Netezza
Teradata
Vertica
Oracle
SQL Server

→

Amazon Redshift

SCTの使用用途



Java, C++, C#等

評価レポートの作成

何割のオブジェクトが自動変換可能かなどのPDFレポートを数クリックで作成でき、変換工数の事前見積もりを補助

スキーマの移行

テーブルやプロシージャなどのデータベースオブジェクトを自動変換しターゲットのデータベースに適用。

アプリケーションソースコードの評価、移行

アプリケーションソースコードに対する評価レポートの作成やソースコード内のSQL変換も可能

AWS Schema Conversion Tool

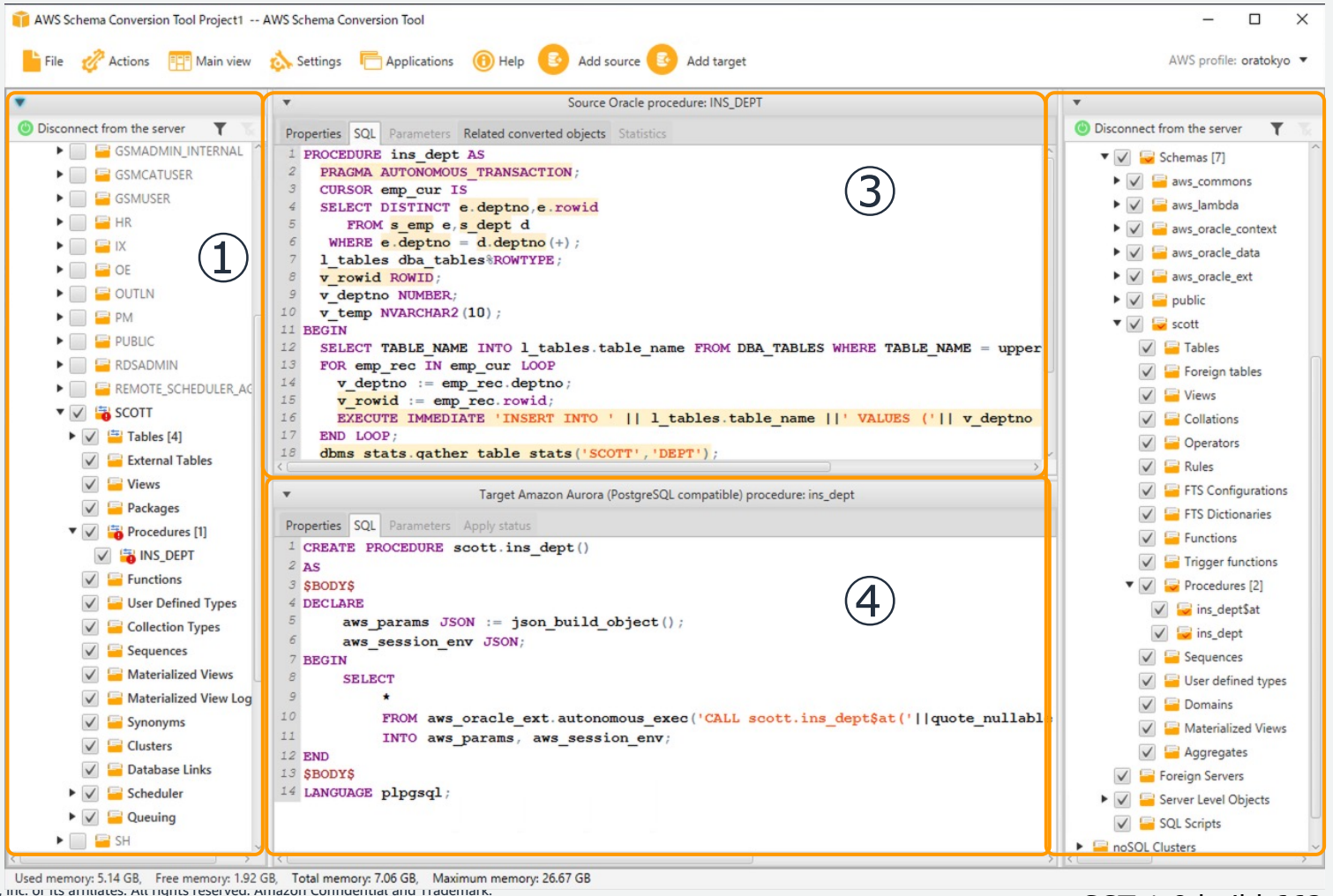
Modernize & Migrate

- 対応OS
  - Fedora Linux
  - Microsoft Windows
  - Ubuntu Linux 15.04
- SCTのインストーラー
  - 以下からダウンロード
    - https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Installing.html

AWS Schema Conversion Tool の基本的な使い方

メインビュー：スキーマ／コード変換

- Sequences
- User-defined types
- Synonyms
- Packages
- Stored procedures
- Functions
- Triggers
- Schemas
- Tables
- Indexes
- Views
- Sort and distribution keys

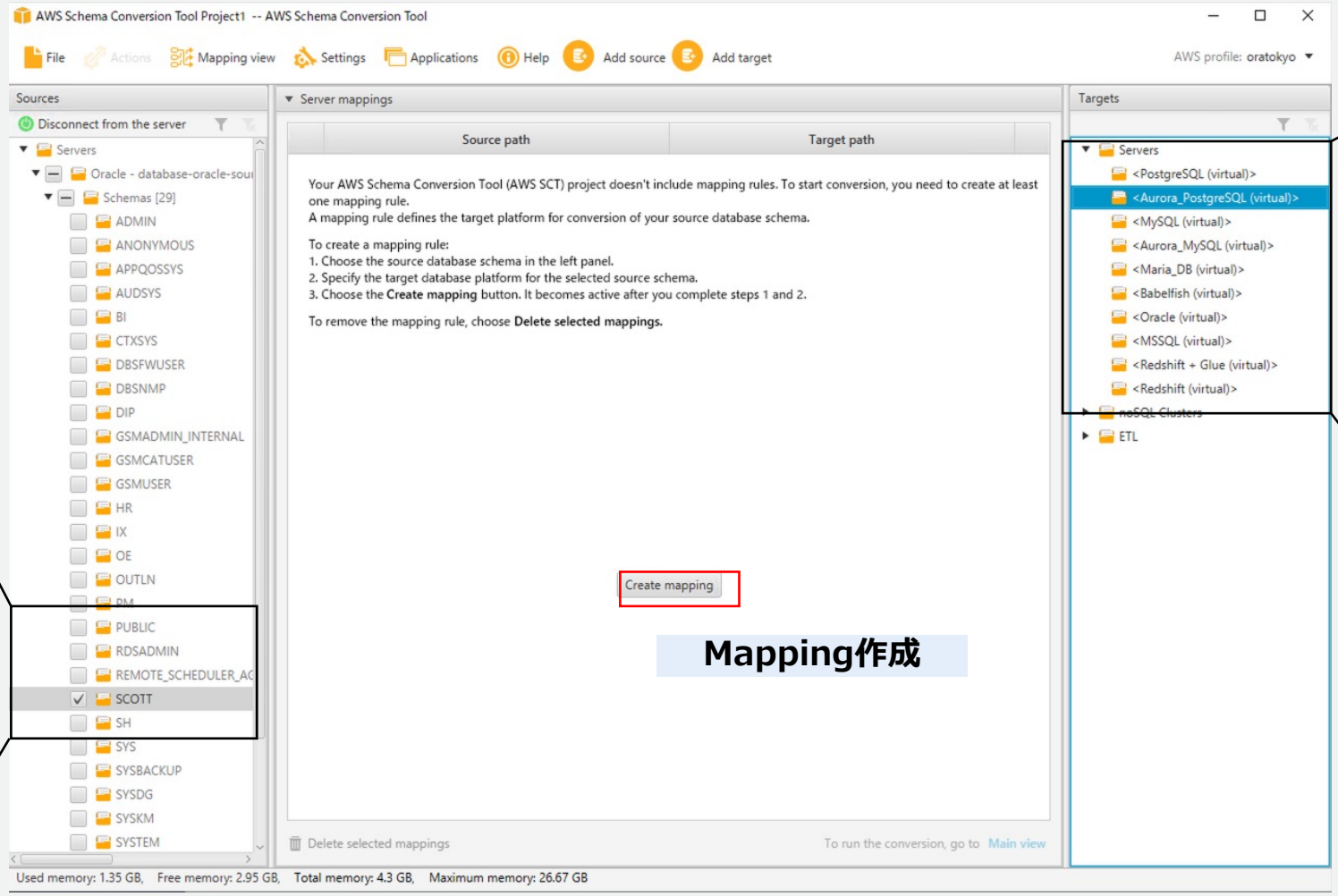


1. Source schema
2. Target schema
3. Source Detail Information
4. Target Information

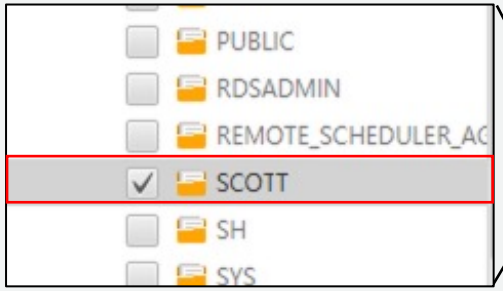
SCT 1.0 build 662

マッピングビュー

移行元と移行先をマッピング

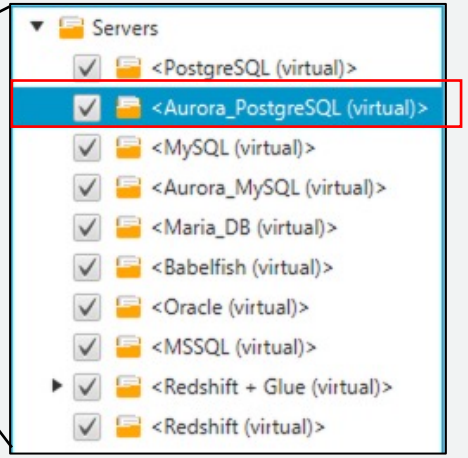


移行元スキーマを指定



Mapping作成

移行先DBを指定



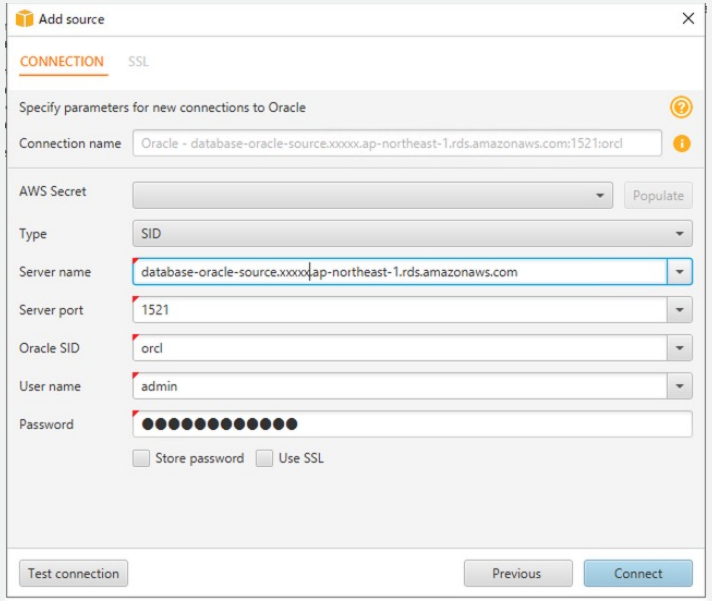
評価レポートは、移行先DBが存在しなくても作成可能（virtualを選択）

SCT 1.0 build 662

Assessment | Schema conversion | Code conversion | Data migration | Validation

評価レポートの作成

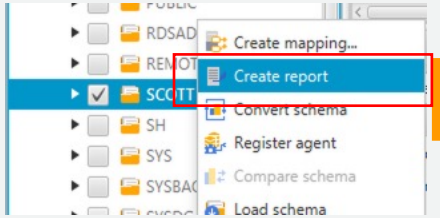
1. ソースDBに接続

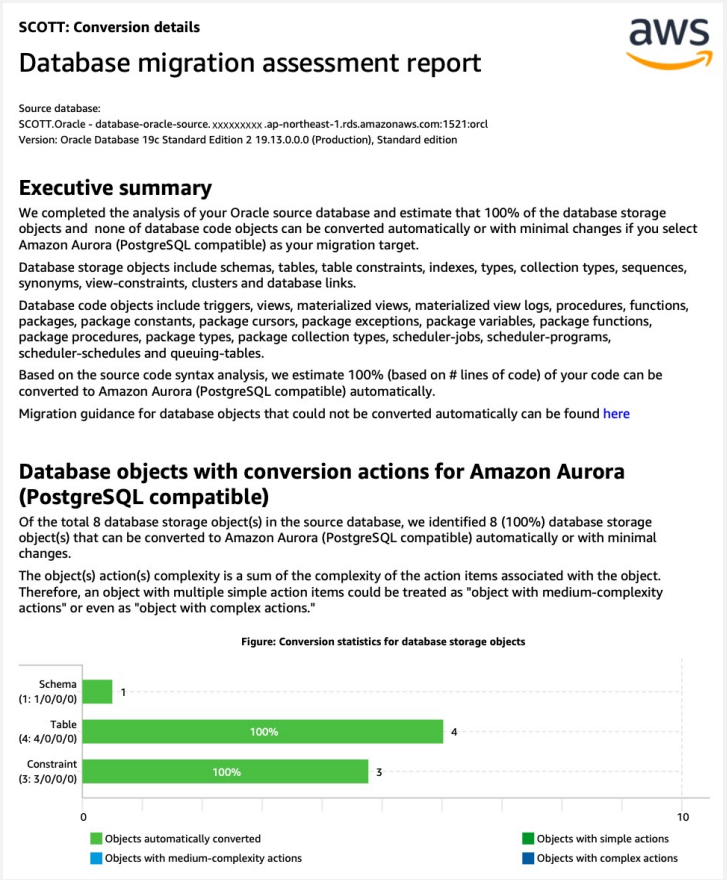


サーバー名
ポート
SID
ユーザー名
パスワード
などを指定

2. create reportで評価レポートを作成

ソース側の
対象スキーマで
Create Reportを選択





SCOTT: Conversion details

Database migration assessment report

Source database:
SCOTT.Oracle - database-oracle-source.xxxxxxxxx.ap-northeast-1.rds.amazonaws.com:1521:orcl
Version: Oracle Database 19c Standard Edition 2 19.13.0.0.0 (Production), Standard edition

Executive summary

We completed the analysis of your Oracle source database and estimate that 100% of the database storage objects and none of database code objects can be converted automatically or with minimal changes if you select Amazon Aurora (PostgreSQL compatible) as your migration target.

Database storage objects include schemas, tables, table constraints, indexes, types, collection types, sequences, synonyms, view-constraints, clusters and database links.

Database code objects include triggers, views, materialized views, materialized view logs, procedures, functions, packages, package constants, package cursors, package exceptions, package variables, package functions, package procedures, package types, package collection types, scheduler-jobs, scheduler-programs, scheduler-schedules and queuing-tables.

Based on the source code syntax analysis, we estimate 100% (based on # lines of code) of your code can be converted to Amazon Aurora (PostgreSQL compatible) automatically.

Migration guidance for database objects that could not be converted automatically can be found here

Database objects with conversion actions for Amazon Aurora (PostgreSQL compatible)

Of the total 8 database storage object(s) in the source database, we identified 8 (100%) database storage object(s) that can be converted to Amazon Aurora (PostgreSQL compatible) automatically or with minimal changes.

The object(s) action(s) complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple action items could be treated as "object with medium-complexity actions" or even as "object with complex actions."

Figure: Conversion statistics for database storage objects

SCTの評価レポート(1/2)

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評価レポート

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

変換工数の事前見積もりを補助

- 自動変換
- 簡単なアクションが必要
- 複雑なアクションが必要
- 大規模なアクションが必要

CTXSYS: Conversion details

Database migration assessment report

Source database:
CTXSYS.Oracle - database-oracle-source XXXXXXXXX .ap-northeast-1.rds.amazonaws.com:1521:orcl
Version: Oracle Database 19c Standard Edition 2 19.13.0.0.0 (Production), Standard edition

Executive summary

We completed the analysis of your Oracle source database and estimate that 97% of the database storage objects and 97% of database code objects can be converted automatically or with minimal changes if you select Amazon Aurora (PostgreSQL compatible) as your migration target.

Database storage objects include schemas, tables, table constraints, indexes, types, collection types, sequences, synonyms, view-constraints, clusters and database links.

Database code objects include triggers, views, materialized views, materialized view logs, procedures, functions, packages, package constants, package cursors, package exceptions, package variables, package functions, package procedures, package types, package collection types, scheduler-jobs, scheduler-programs, scheduler-schedules and queuing-tables.

Based on the source code syntax analysis, we estimate 97% (based on # lines of code) of your code can be converted to Amazon Aurora (PostgreSQL compatible) automatically.

To complete the migration, we recommend 288 conversion action(s) ranging from simple tasks to medium-complexity actions to complex conversion actions.

Migration guidance for database objects that could not be converted automatically can be found here

Database objects with conversion actions for Amazon Aurora (PostgreSQL compatible)

Of the total 146 database storage object(s) and 1,517 database code object(s) in the source database, we identified 141 (97%) database storage object(s) and 1,471 (97%) database code object(s) that can be converted to Amazon Aurora (PostgreSQL compatible) automatically or with minimal changes.

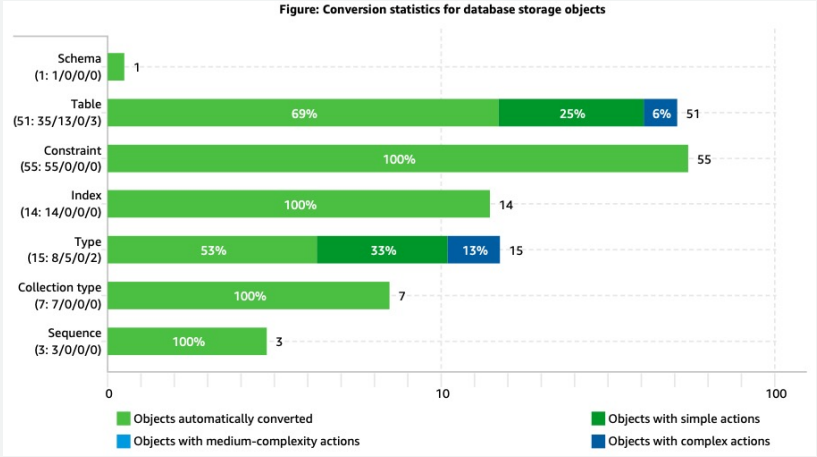
We found 75 encrypted object(s).

5 (3%) database storage object(s) require 5 complex user action(s) to complete the conversion.

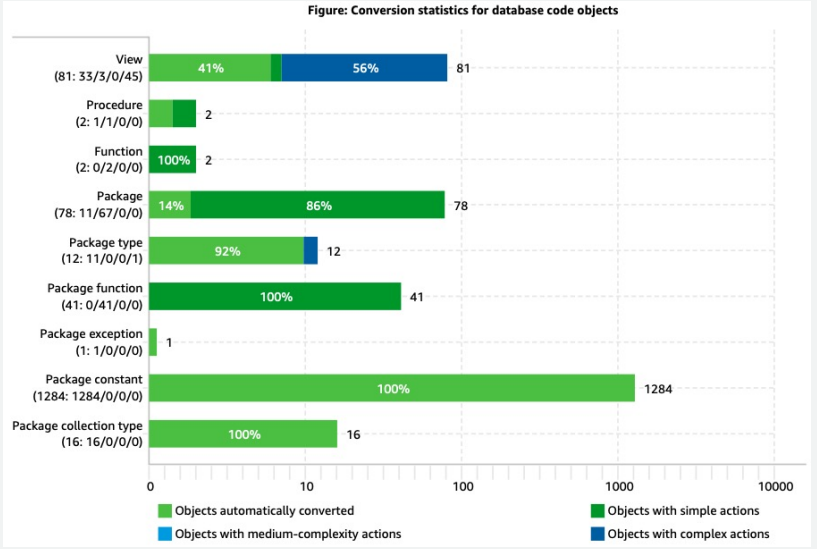
46 (3%) database code object(s) require 97 complex user action(s) to complete the conversion.

The object(s) action(s) complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple action items could be treated as "object with medium-complexity actions" or even as "object with complex actions."

Conversation Statistics for database storage objects



Conversation Statistics for database code objects



SCTの評価レポート(2/2)

自動変換ができない場合は詳細レベルの原因や推奨アクションを提示

自動変換できないオブジェクトの情報を一覧で確認したい場合、CSVに出力することも可能

CTXSYS: Conversion details

Source database:
CTXSYS.Oracle - database-oracle-source.cicxfujxfrr7.ap-northeast-1.rds.amazonaws.com:1521:orcl
Version: Oracle Database 19c Standard Edition 2 19.13.0.0.0 (Production), Standard edition

Schemas.CTXSYS.Tables.DR$INDEX_CDI_COLUMN
Schemas.CTXSYS.Tables.DR$INDEX_OBJECT
Schemas.CTXSYS.Tables.DR$ONLINE_PENDING
+11 more

Issue 5550: PostgreSQL doesn't have a data type ROWID
Recommended action: Review your transformed code and modify it if necessary.
Issue code: 5550 | Number of occurrences: 3 | Estimated complexity: Complex
Documentation references: http://www.postgresql.org/docs/current/static/datatype.html
Schemas.CTXSYS.Tables.DR$PENDING.Columns.PND_ROWID
Schemas.CTXSYS.Tables.DR$UNINDEXED.Columns.UNX_ROWID
Schemas.CTXSYS.Tables.DR$WAITING.Columns.WTG_ROWID

Type Changes
Not all types can be converted automatically. You'll need to address these issues manually.

Issue 5582: Encrypted Objects
Recommended action: Decrypt the object before conversion.
Issue code: 5582 | Number of occurrences: 5 | Estimated complexity: Simple
Schemas.CTXSYS.User Defined Types.CATINDEXMETHODS
Schemas.CTXSYS.User Defined Types.RULEINDEXMETHODS
Schemas.CTXSYS.User Defined Types.TEXTINDEXMETHODS
Schemas.CTXSYS.User Defined Types.TEXTOPTSTATS
Schemas.CTXSYS.User Defined Types.XPATHINDEXMETHODS

Issue 5028: Unable to convert unsupported datatypes
Recommended action: The stub-code created. Please perform a manual conversion.
Issue code: 5028 | Number of occurrences: 1 | Estimated complexity: Complex
Schemas.CTXSYS.User Defined Types.DR$SESSION_STATE_T.Attributes.DUMPEDERRORS

Issue 5572: PostgreSQL doesn't support object type methods
Recommended action: Revise your architecture with a custom solution for the user type.
Issue code: 5572 | Number of occurrences: 1 | Estimated complexity: Complex
Documentation references: http://www.postgresql.org/docs/current/static/sql-createtype.html
Schemas.CTXSYS.User Defined Types.CTX_FEEDBACK_ITEM_TYPE

Code object actions

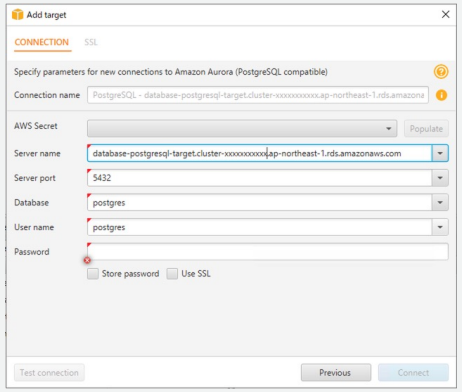
View Changes
Not all views can be converted automatically. You'll need to address these issues manually.

Amazon Aurora (PostgreSQL compatible)_3cAurora_PostgreSQL (virtual)3e

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Category	Occurrence	Action item	Subject	Group	Description	Documentat	Recommend	Filtered	Estimated co	Line	Position	Source	Target
2	Package typ	Schemas.CT	5028	CTXSYS.DR_	Unable to co	Unable to co	null	The stub-code created. P		Complex			Oracle - data	<Aurora_PostgreSQL (virtual)>
3	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
4	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
5	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
6	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
7	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
8	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
9	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
10	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
11	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
12	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
13	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
14	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
15	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
16	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
17	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
18	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
19	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
20	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
21	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
22	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
23	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
24	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
25	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>
26	Package fun	Schemas.CT	5027		Unable to co	Unable to co	null	Please check if package s		Simple			Oracle - data	<Aurora_PostgreSQL (virtual)>

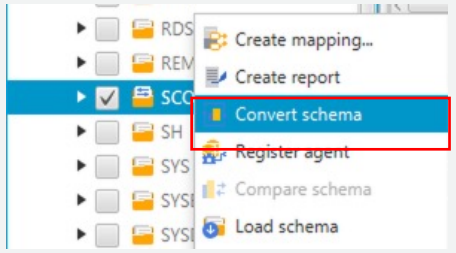
ターゲットDBにスキーマ作成

1. ターゲットDBに接続



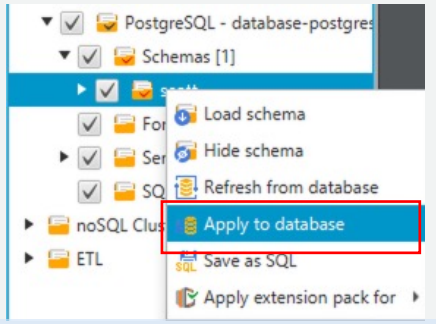
サーバー名
ポート
SID
ユーザー名
パスワード
などを指定

2. スキーマを移行



ターゲット側の
対象スキーマで
Convert Schemaを選択

3. ターゲットDBに適用



Apply to Databaseで
移行先にスキーマ作成

```
postgres=> ¥dt
          List of relations
 Schema |   Name   | Type  |  Owner
--------+----------+-------+----------
 scott  | bonus    | table | postgres
 scott  | dept     | table | postgres
 scott  | emp      | table | postgres
 scott  | salgrade | table | postgres
(4 rows)
```

ターゲットDBの
オブジェクト

スキーマの移行（プロシージャの変換例）

Oracle PL/SQL

```
CREATE OR REPLACE PROCEDURE ins_dept AS
  CURSOR emp_cur IS
  SELECT DISTINCT e.deptno
    FROM emp e,dept d
   WHERE e.deptno = d.deptno(+);

BEGIN
  FOR emp_rec IN emp_cur LOOP
    INSERT INTO dept
VALUES(emp_rec.deptno,'XXX',null);
  END LOOP;
EXCEPTION
 WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

宣言部

実行部

例外処理部

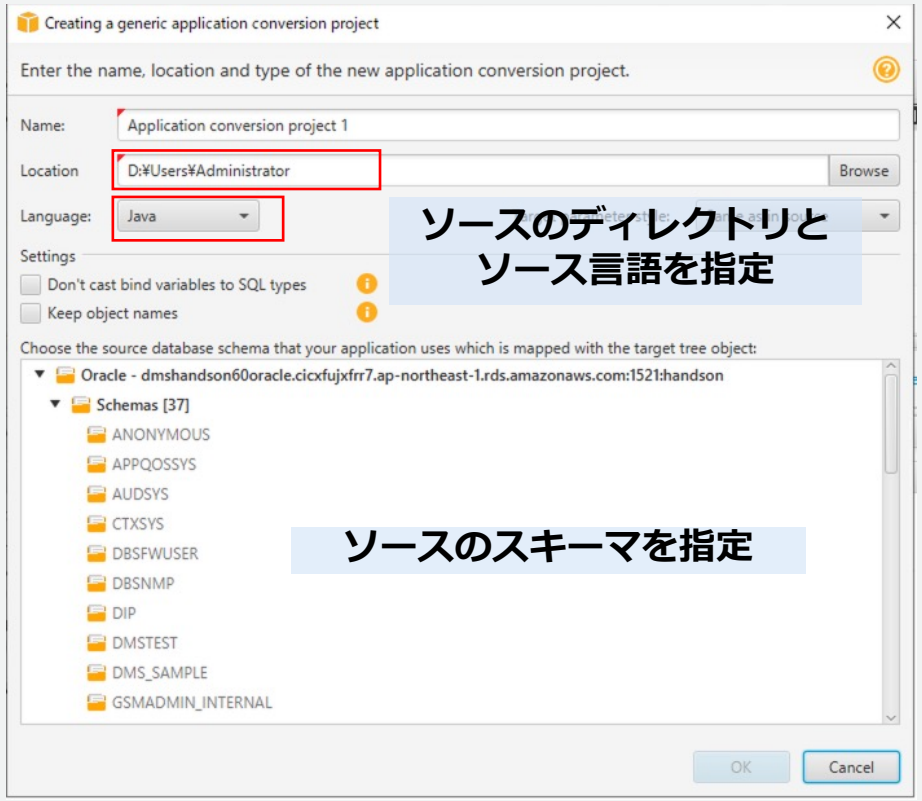
PostgreSQL PL/pgSQL

```
CREATE OR REPLACE PROCEDURE scott.ins_dept() AS
$BODY$
DECLARE
   emp_cur CURSOR FOR
   SELECT DISTINCT e.deptno
      FROM scott.emp AS e
      LEFT OUTER JOIN scott.dept AS d
         ON (e.deptno = d.deptno);
   emp_cur$ATTRIBUTES
aws_oracle_data.TCursorAttributes := ROW (FALSE, NULL,
NULL, NULL);
BEGIN
   FOR emp_rec IN emp_cur LOOP
      INSERT INTO scott.dept
      VALUES (emp_rec.deptno, 'XXX', NULL);
   END LOOP;
   EXCEPTION
      WHEN others THEN
   RAISE DEBUG USING MESSAGE = SQLERRM::TEXT;
END;
$BODY$
LANGUAGE plpgsql;
```

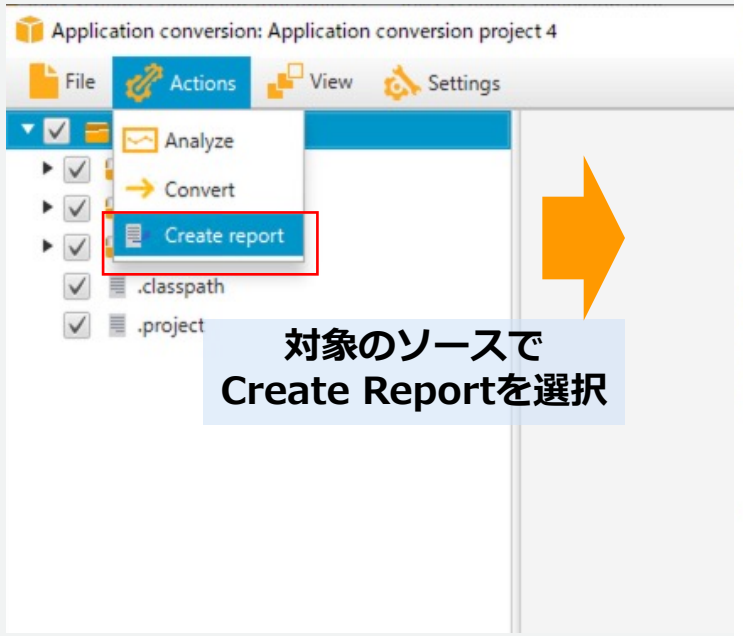

SCT 1.0 build 662

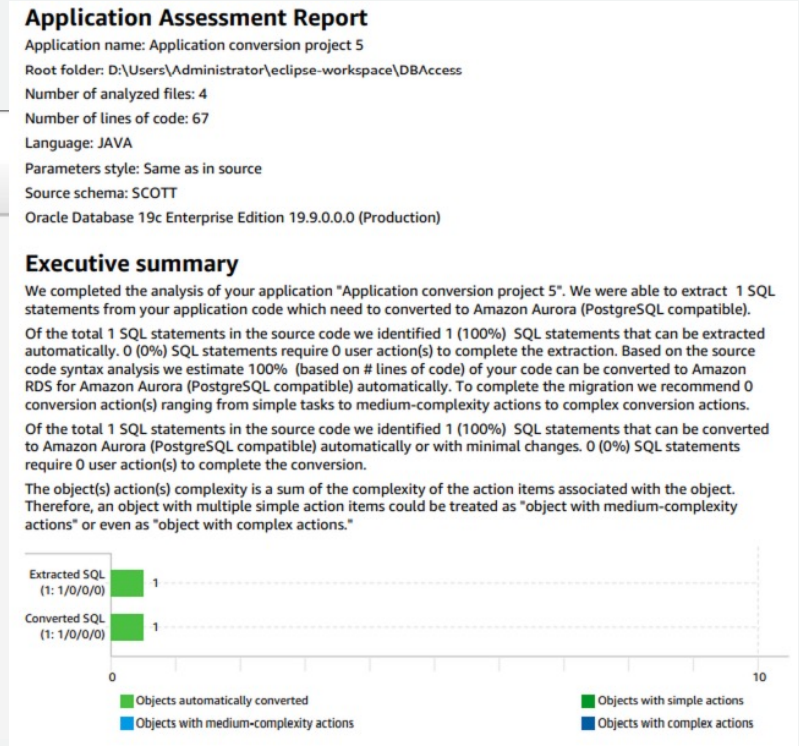
アプリケーションソースコードの評価

1. ソースコードとスキーマを選択

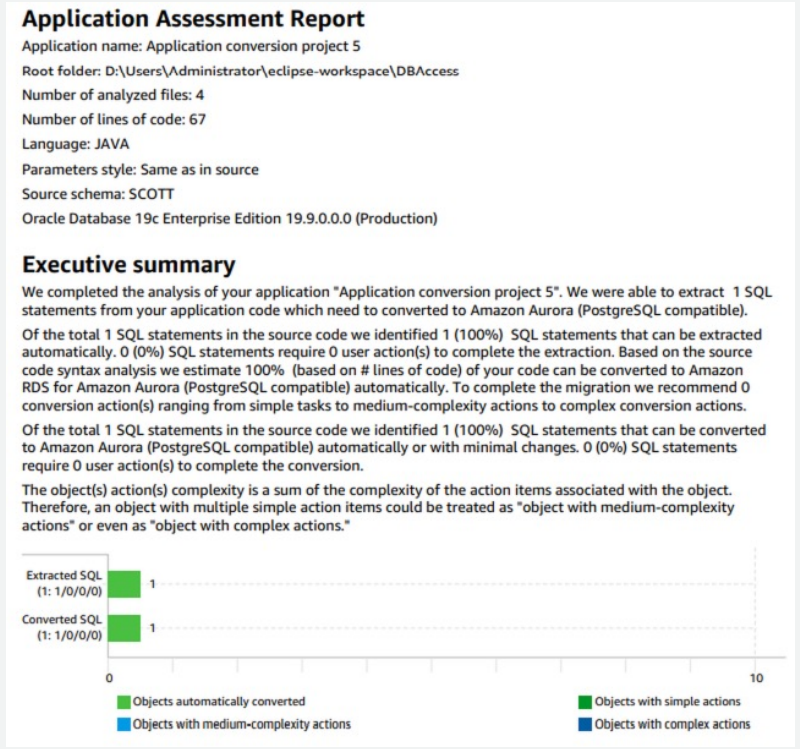


2. 評価レポートを作成





SCTの評価レポート（アプリケーション）



Application Assessment Report

Application name: Application conversion project 5

Root folder: D:\Users\Administrator\eclipse-workspace\DBAccess

Number of analyzed files: 4

Number of lines of code: 67

Language: JAVA

Parameters style: Same as in source

Source schema: SCOTT

Oracle Database 19c Enterprise Edition 19.9.0.0.0 (Production)

Executive summary

We completed the analysis of your application "Application conversion project 5". We were able to extract 1 SQL statements from your application code which need to converted to Amazon Aurora (PostgreSQL compatible).

Of the total 1 SQL statements in the source code we identified 1 (100%) SQL statements that can be extracted automatically. 0 (0%) SQL statements require 0 user action(s) to complete the extraction. Based on the source code syntax analysis we estimate 100% (based on # lines of code) of your code can be converted to Amazon RDS for Amazon Aurora (PostgreSQL compatible) automatically. To complete the migration we recommend 0 conversion action(s) ranging from simple tasks to medium-complexity actions to complex conversion actions.

Of the total 1 SQL statements in the source code we identified 1 (100%) SQL statements that can be converted to Amazon Aurora (PostgreSQL compatible) automatically or with minimal changes. 0 (0%) SQL statements require 0 user action(s) to complete the conversion.

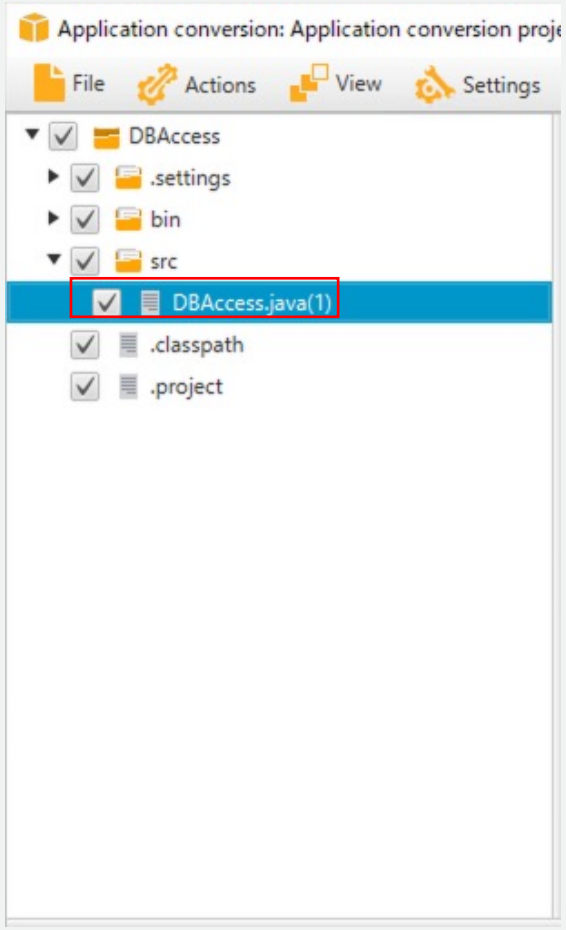
The object(s) action(s) complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple action items could be treated as "object with medium-complexity actions" or even as "object with complex actions."

Extracted SQL：SCTがソースからSQLを抽出した結果
→SQLの可能性があるものを抽出し難易度を評価
Converted SQL：抽出されたSQLを変換した結果
→Extracted SQLからSCTがターゲットのDBに変換し、その難易度を評価

Extracted code	Converted code
"select ¥n" + " empno, rpad(ename,10) || 'JOB(' || job || ')' name ¥n" + " from emp ¥n" + " where empno = ?"	SELECT empno, CONCAT_WS('', RPAD(ename, 10), 'JOB(', job, ')') AS name FROM scott.emp WHERE empno = (?)::NUMERIC

アプリケーションソースコードの変換

1. ソースを選択



2. ソースを変換

ソースの移行（Javaソースの変換例）

Java(Oracle)

```
import java.sql.*;

public class DBAccess {
  public static void main (String args[])
  throws SQLException, ClassNotFoundException {
    Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@xxxxxx.ap-
northeast-1.rds.amazonaws.com:1521/handson", "scott", "tigertiger");
    String selectstatement =
          "select ¥n" +
          " empno, rpad(ename,10) || 'JOB(' || job || ')' name ¥n" +
          " from emp ¥n" +
          " where empno = ?";
PreparedStatement prepstmt = conn.prepareStatement(selectstatement);
    prepstmt.setInt(1, 7934);
    ResultSet rset = prepstmt.executeQuery();

    while ( rset.next() ) {
      System.out.println(rset.getInt(1) + "¥t" + rset.getString(2));
        }
    rset.close();
    prepstmt.close();
    conn.close();
  }
}
```

接続設定は手動変換

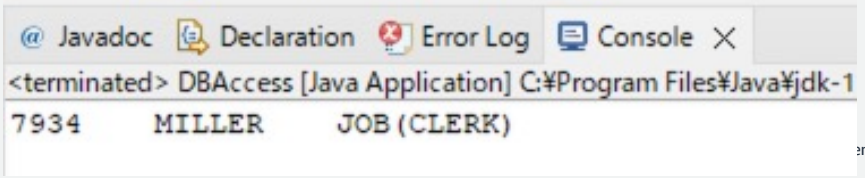
SCTによる変換

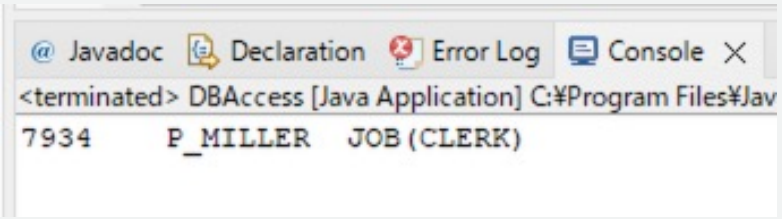
PostgreSQL PL/pgSQL

```
import java.sql.*;

public class DBAccess {
  public static void main (String args[])
  throws SQLException, ClassNotFoundException {
    Connection conn = DriverManager.getConnection("jdbc:postgresql://xxxxxxx.ap-
northeast-1.rds.amazonaws.com:5432/postgres", "postgres", "Insight13110");
    String selectstatement =
          "SELECT¥n"+
          "   empno, CONCAT_WS('', RPAD(ename, 10), 'JOB(', job, ')') AS name¥n"+
          "   FROM scott.emp¥n"+
          "   WHERE empno = (?)::NUMERIC";
PreparedStatement prepstmt = conn.prepareStatement(selectstatement);
    prepstmt.setInt(1, 7934);
    ResultSet rset = prepstmt.executeQuery();

    while ( rset.next() ) {
      System.out.println(rset.getInt(1) + "¥t" + rset.getString(2));
        }
    rset.close();
    prepstmt.close();
    conn.close();
  }
}
```





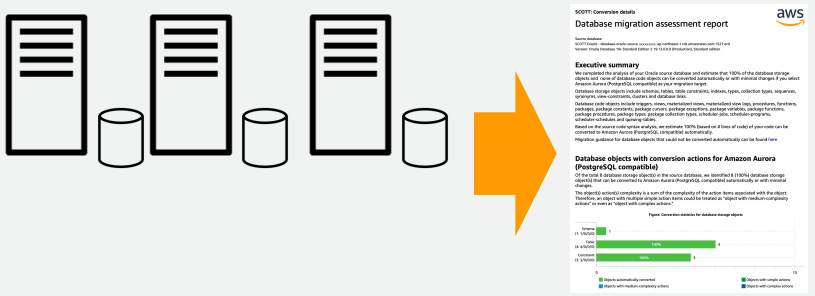
AWS Schema Conversion Tool ちょっぴりDive Deep

AWS Schema Conversion Tool ちょっぴりDive Deep

- マルチサーバー評価
- 移行工数の見積
- 自動変換されない原因調査方法
- アプリケーションソースコード評価時の注意

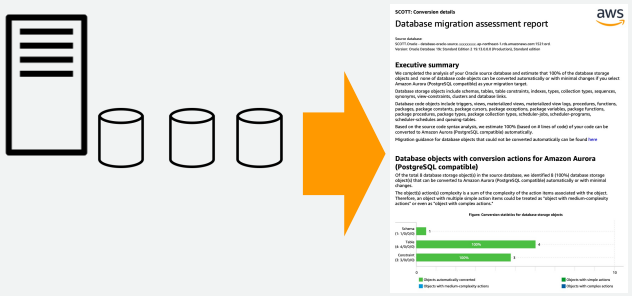
マルチサーバー評価

1. 複数サーバーの評価



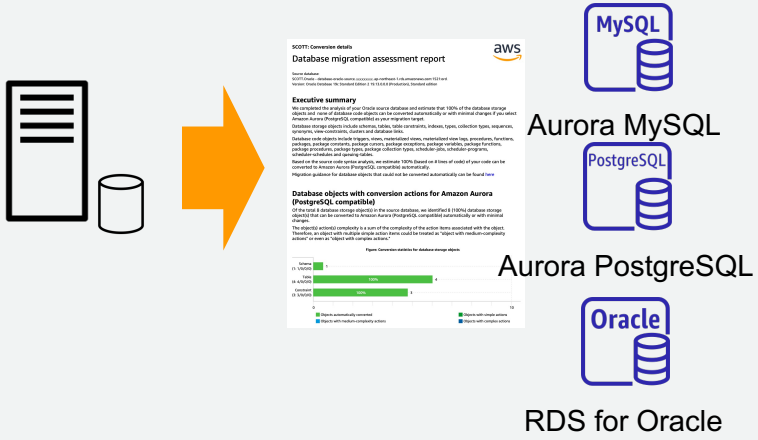
移行対象検討時に複数サーバーを指定

2. 複数スキーマの評価



１つのシステムで複数スキーマを使用

3. 複数ターゲットの評価



ターゲットのDBエンジンの選択

移行工数の見積

CSVには、変換エラーの一覧が出力されます。この変換エラー毎にSCTが算出した移行工数が出力されています。このSCTが算出した移行工数を計算することで、大凡の移行工数を算出することができます。

Schema	Action item	Number of occurrences (A)	Learning curve efforts (B)	Efforts to convert an occurrence of the action item (C)
USERDB	5027	41	1	0.5
USERDB	5028	7	40	8
USERDB	5103	14	40	0.1
USERDB	5340	42	80	16
USERDB	5550	11	0	8
USERDB	5572	1	40	40
USERDB	5578	40	0	8
USERDB	5581	16	4	0.1
USERDB	5582	75	0	2
USERDB	5591	40	40	0.01
USERDB	5651	1	8	8

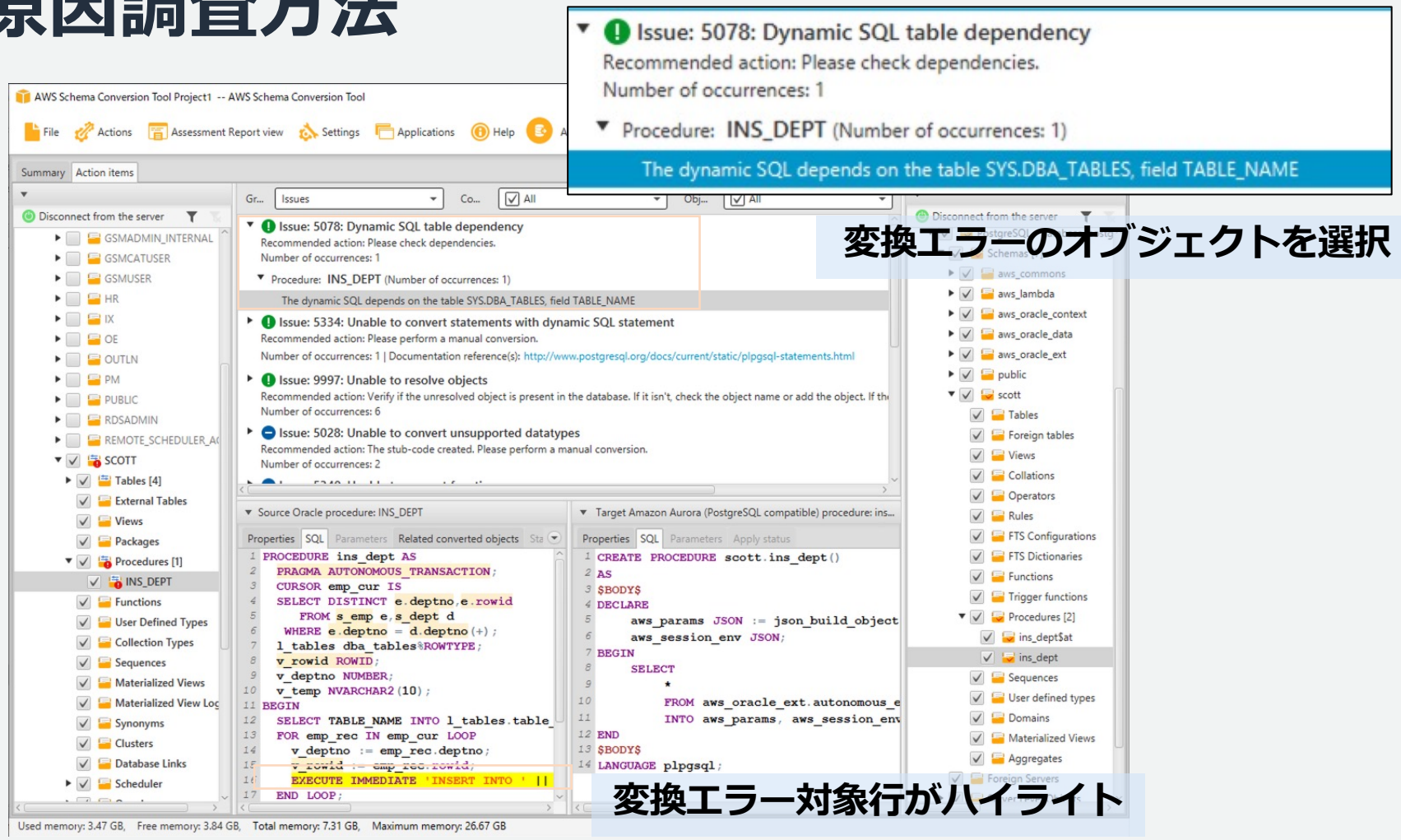
Number of occurrences(A): Action Itemの発生回数
Learning curve efforts(B): 問題の改修方法を決定するのに要する時間
Efforts to convert an occurrence of the action item(C): 問題を修正する時間

Action Item毎の工数(時間) = (B) + (A) x (C)　全体の移行工数(時間) = 全てのAction Itemの工数の合計

自動変換されない原因調査方法

変換エラーに表示されたオブジェクトを選択すると、対象行がハイライトされる

変換エラーメッセージや詳細を参照しながら、変換エラーの原因を調査



変換エラー例

```
CREATE OR REPLACE PROCEDURE ins_dept AS
  PRAGMA AUTONOMOUS_TRANSACTION;
  CURSOR emp_cur IS
  SELECT DISTINCT e.deptno,e.rowid
     FROM s_emp e,s_dept d
    WHERE e.deptno = d.deptno(+);
  l_tables dba_tables%ROWTYPE;
  v_rowid ROWID;
  v_deptno NUMBER;
  v_temp NVARCHAR2(10);
BEGIN
  SELECT TABLE_NAME INTO l_tables.table_name FROM DBA_TABLES WHERE TABLE_NAME = upper('dept');
  FOR emp_rec IN emp_cur LOOP
    v_deptno := emp_rec.deptno;
    v_rowid := emp_rec.rowid;
   EXECUTE IMMEDIATE 'INSERT INTO ' || l_tables.table_name ||' VALUES ('|| v_deptno ||',''XXX'',' ||
v_rowid || ')';
  END LOOP;
  dbms_stats.gather_table_stats('SCOTT','DEPT');
EXCEPTION
 WHEN PROGRAM_ERROR THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
 WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

Issue: 5610 : This statement references to a synonym

Cause

シノニムが使用されている場合、このメッセージが出力される
(PostgreSQLにシノニムが存在しない)
このスクリプトでは、s_empとs_deptでシノニムを使用。

Action

SCTではシノニムの元になるオブジェクト名が指定される為無視可能

Issue: 5334 : Unable to convert statements with dynamic SQL statement

Cause

動的にSQL文を作成して実行する処理の場合、このメッセージが出力される
（動的SQL自体は使用可能）

Action

動的に作成されたSQL文がPostgreSQLで実行可能かどうかを確認

Issue: 5078 : Dynamic SQL table dependency

Cause

動的にSQL文を生成していて、SQL文の中で別テーブルから取得した値を使用している場合、このメッセージが出力される

Action

別テーブルの情報が適切に取得されること、取得した値でSQLが実行できることを確認

アプリケーションソースコード評価時の注意

以下のようなSQLは変換可能なSQLとして抽出されない

SELECT、FROM、WHERE句を個別に定義

```
String selectstatement =
      "select ¥n" +
      " empno, rpad(ename,10) || 'JOB(' || job || ')' name ¥n" ;
String fromstatement =
      " from emp ¥n" ;
String wherestatement =
      " where empno = ?";
String sqlstatement =
      selectstatement + fromstatement + wherestatement;
```

Extracted code	Converted code
"select ¥n" + " empno, rpad(ename,10) || 'JOB(' || job || ')' name ¥n"	

クエリ内の項目を変数で定義

```
String tablename =
      ”emp;
String selectstatement =
      "select ¥n" +
      " empno, rpad(ename,10) || 'JOB(' || job || ')' name ¥n" +
      " from " + tablename + " ¥n" +
      " where empno = ?";
```

Extracted code	Converted code
"select ¥n" + " empno, rpad(ename,10) || 'JOB(' || job || ')' name ¥n" + " from " + tablename + " ¥n" + " where empno = ?"	

まとめ

まとめ

- AWS Schema Conversion Toolを使用することで、データベースエンジン間の移行が容易に
- スキーマやアプリケーションの移行難易度や移行工数を算出
- スキーマやソースコードの変更を適用して移行工数を削減

Thank you!