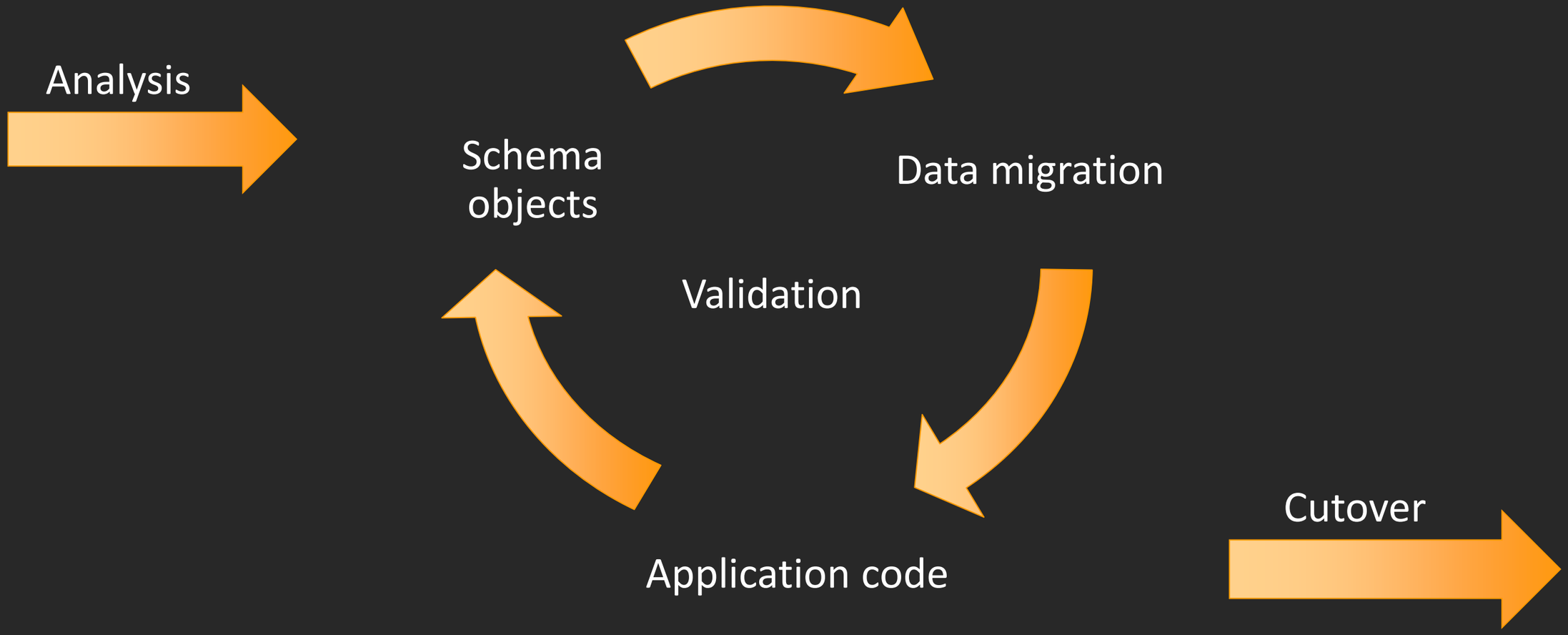
AWS 데이터베이스 서비스로 마이그레이션하기

Jihoon Kim, Database Specialist Solution Architect
Dec, 2021

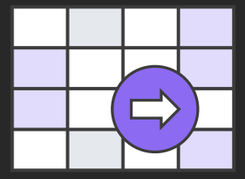
A typical database migration lifecycle



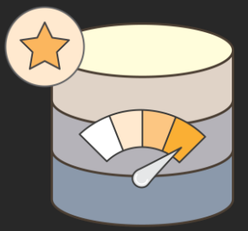
Database migration process

Step 1

Source
Oracle Database on-premises, Amazon EC2, or Amazon RDS



AWS Schema Conversion Tool (AWS SCT)

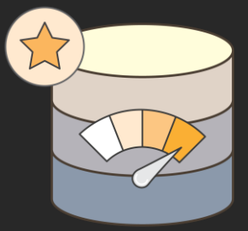


Target
Amazon Aurora

Step 2

Source
Oracle Database on-premises, Amazon EC2, or Amazon RDS

AWS Database Migration Service (AWS DMS)



Target
Amazon Aurora

Oracle to Amazon Aurora migration playbook

- In-depth guidance on Oracle to Aurora PostgreSQL migration
- With detailed examples
- Migration best practices

Schema + Data + Best practices

AWS SCT + AWS DMS + Playbook

	Oracle Feature	PostgreSQL Feature	Compatibility
Link	Index Organized Tables (IOTs)	PostgreSQL "Cluster" Tables	Yes*
Link	Common Data Types	Common Data Types	Yes
Link	Table Constraints	Table Constraints	Yes
Link	Table Partitioning including: RANGE, LIST, HASH, COMPOSITE, Automatic LIST	Table Partitioning including: RANGE, LIST	Yes*
Link	Exchange & Split Partitions	N/A	None
Link	Temporary Tables	Temporary Tables	Yes*
Link	Unused Columns	ALTER TABLE DROP COLUMN	Yes
Link	Virtual Columns	Views and/or Function as a Column	Yes*
Link	User Defined Types (UDTs)	User Defined Types (UDTs)	Yes
Link	Read Only Tables & Table Partitions	Read Only Roles and/or Triggers	Yes*
Link	Index Typ		
Link	B-Tree Inc		
Link	Composit		
Link	BITMAP Ir		
Link	Function-		
Link	Global an Indexes		
Link	Identity C		
Link	MVCC (Table & F		
Link	Character		
Link	Transactic		
Link	Recovery Manager (RMAN)	AWS Aurora Snapshots	Yes
Link	Flashback Database	AWS Aurora Snapshots	Yes
Link	12c Multi-tenant architecture: PDBs and CDB	Databases	Yes*
Link	Tablespaces & DataFiles	Tablespaces	Yes*
Link	Data Pump	pg_dump & pg_restore	Yes
Link	Resource Manager	Separate AWS Aurora Clusters	Yes
Link	Database Users	Database Roles	Yes
Link	Database Roles	Database Roles	Yes
Link	SGA & PGA Memory	Memory Buffers	Yes
Link	V$ Views & Data Dictionary	System Catalog Tables, Statistics Collector, AWS Aurora Performance Insights	Yes*
Link	Log Miner	Logging Options	Yes
Link	Instance & Database Parameters (SPFILE)	AWS Aurora Parameter Groups	Yes
Link	Session Parameters	Session Parameters	Yes
Link	Alert.log (error log)	Error Log via AWS Console	Yes
Link	Automatic and Manual Statistics Collection	Automatic and Manual Statistics Collection	Yes
Link	Viewing Execution Plans	Viewing Execution Plans	Yes

https://d1.awsstatic.com/whitepapers/Migration/oracle-database-amazon-aurora-postgresql-migration-playbook-12.4.pdf

aws

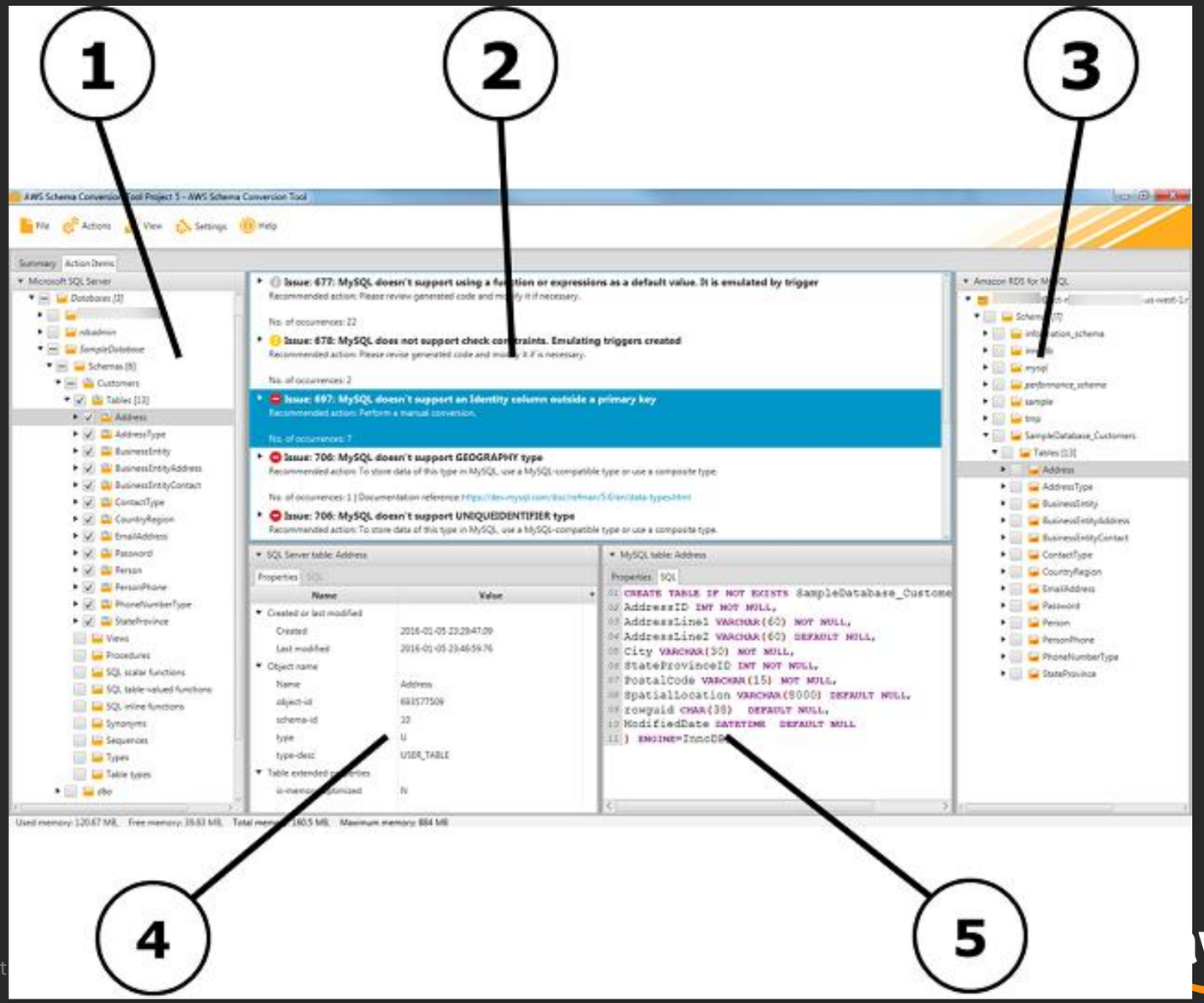
AWS Schema Conversion Tool (AWS SCT)

The AWS Schema Conversion Tool helps automate many database schema and code conversion tasks when migrating from source to target database engines

- Convert database schema
- Convert data warehouse schema
- Convert application SQL

Components of the AWS SCT console

1. Source schema
2. Action items
3. Target schema
4. Schema element details
5. Edit window



AWS SCT can tell you how hard the migration will be

Database Migration Assessment Report

Source Database: [redacted]
Oracle Database 11g 11.2.0.3.0 (64bit Production), Standard edition

Executive Summary

We completed the analysis of your Oracle source database and estimate that 76% of the database storage objects and 65% 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, sequences, synonyms, view-constraints and clusters. 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 and package collection types. Based on our analysis of SQL syntax elements of your source database schema, we estimate that 98% of your entire database schema can be converted to Amazon Aurora (PostgreSQL compatible) automatically. To complete the migration, we recommend 606 conversion action(s) ranging from simple tasks to medium-complexity actions to significant conversion actions.

Database Objects with Conversion Actions for Amazon Aurora (PostgreSQL compatible)

Of the total 327 database storage object(s) and 57 database code object(s) in the source database, we were able to identify 250 (76%) database storage object(s) and 37 (65%) database code object(s) that can be converted to Amazon Aurora (PostgreSQL compatible) automatically or with minimal changes.

77 (24%) database storage object(s) require 77 significant user action(s) to complete the conversion.

20 (35%) database code object(s) require 20 significant user action(s) to complete the conversion.

Figure: Conversion statistics for database storage objects

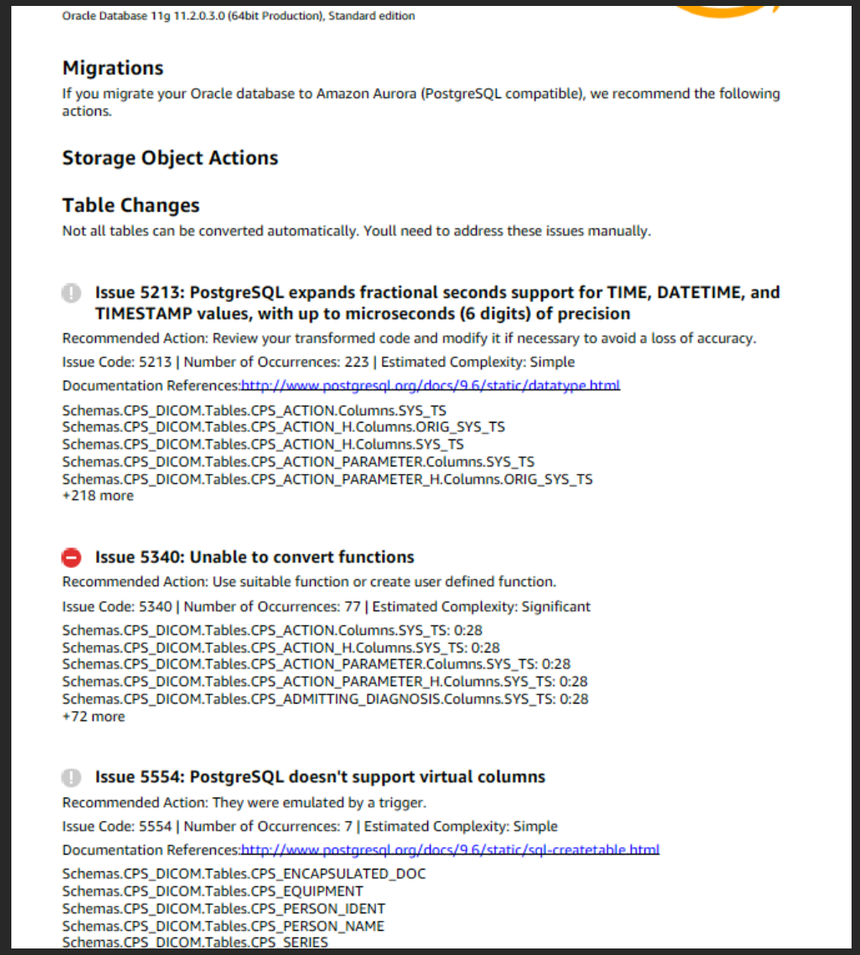
Figure: Conversion statistics for database code objects

Connect AWS SCT to source and target databases

Run assessment report

Read executive summary

Follow detailed instructions



Oracle Database 11g 11.2.0.3.0 (64bit Production), Standard edition

Migrations

If you migrate your Oracle database to Amazon Aurora (PostgreSQL compatible), we recommend the following actions.

Storage Object Actions

Table Changes

Not all tables can be converted automatically. Youll need to address these issues manually.

Issue 5213: PostgreSQL expands fractional seconds support for TIME, DATETIME, and TIMESTAMP values, with up to microseconds (6 digits) of precision

Recommended Action: Review your transformed code and modify it if necessary to avoid a loss of accuracy.

Issue Code: 5213 | Number of Occurrences: 223 | Estimated Complexity: Simple

Documentation References:http://www.postgresql.org/docs/9.6/static/datatype.html

Schemas.CPS_DICOM.Tables.CPS_ACTION.Columns.SYS_TS
Schemas.CPS_DICOM.Tables.CPS_ACTION_H.Columns.ORIG_SYS_TS
Schemas.CPS_DICOM.Tables.CPS_ACTION_H.Columns.SYS_TS
Schemas.CPS_DICOM.Tables.CPS_ACTION_PARAMETER.Columns.SYS_TS
Schemas.CPS_DICOM.Tables.CPS_ACTION_PARAMETER_H.Columns.ORIG_SYS_TS
+218 more

Issue 5340: Unable to convert functions

Recommended Action: Use suitable function or create user defined function.

Issue Code: 5340 | Number of Occurrences: 77 | Estimated Complexity: Significant

Schemas.CPS_DICOM.Tables.CPS_ACTION.Columns.SYS_TS: 0:28
Schemas.CPS_DICOM.Tables.CPS_ACTION_H.Columns.SYS_TS: 0:28
Schemas.CPS_DICOM.Tables.CPS_ACTION_PARAMETER.Columns.SYS_TS: 0:28
Schemas.CPS_DICOM.Tables.CPS_ACTION_PARAMETER_H.Columns.SYS_TS: 0:28
Schemas.CPS_DICOM.Tables.CPS_ADMITTING_DIAGNOSIS.Columns.SYS_TS: 0:28
+72 more

Issue 5554: PostgreSQL doesn't support virtual columns

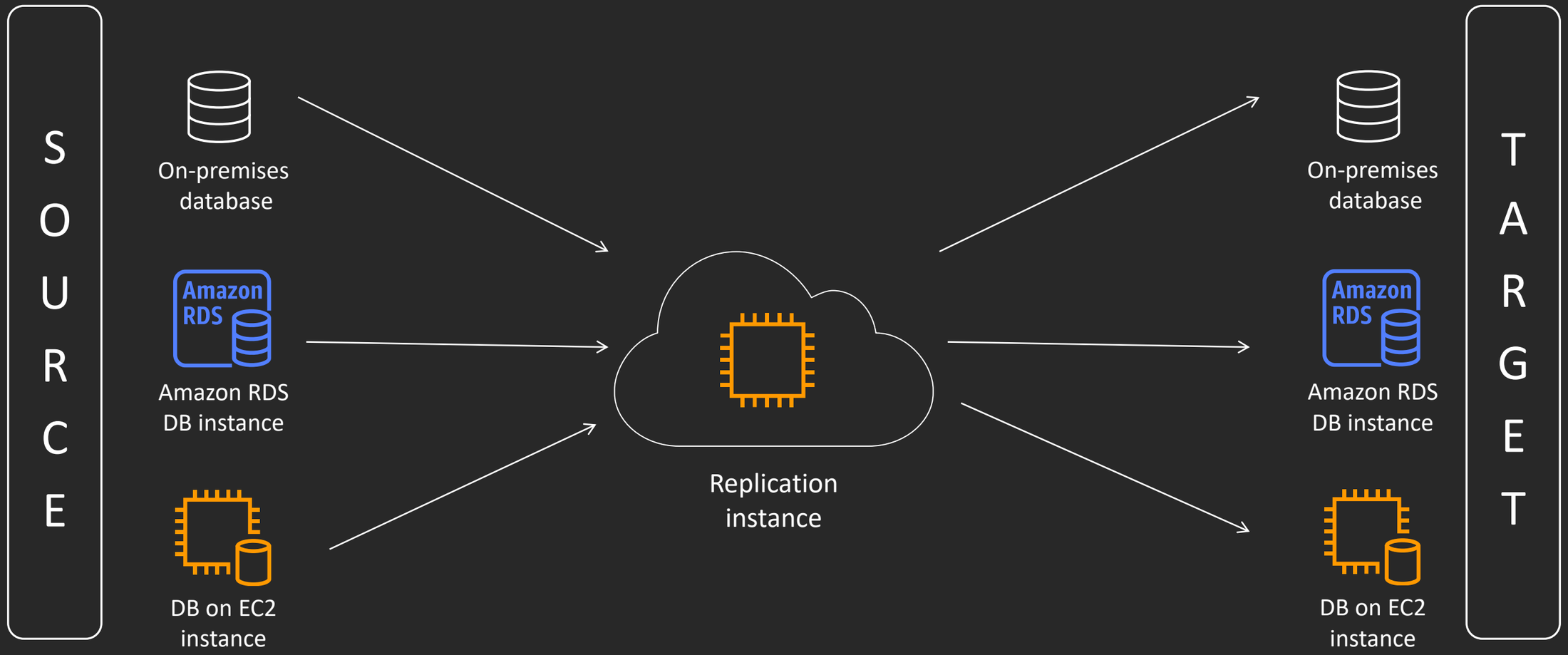
Recommended Action: They were emulated by a trigger.

Issue Code: 5554 | Number of Occurrences: 7 | Estimated Complexity: Simple

Documentation References:http://www.postgresql.org/docs/9.6/static/sql-createtable.html

Schemas.CPS_DICOM.Tables.CPS_ENCAPSULATED_DOC
Schemas.CPS_DICOM.Tables.CPS_EQUIPMENT
Schemas.CPS_DICOM.Tables.CPS_PERSON_IDENT
Schemas.CPS_DICOM.Tables.CPS_PERSON_NAME
Schemas.CPS_DICOM.Tables.CPS_SERIES

AWS Database Migration Service (AWS DMS)



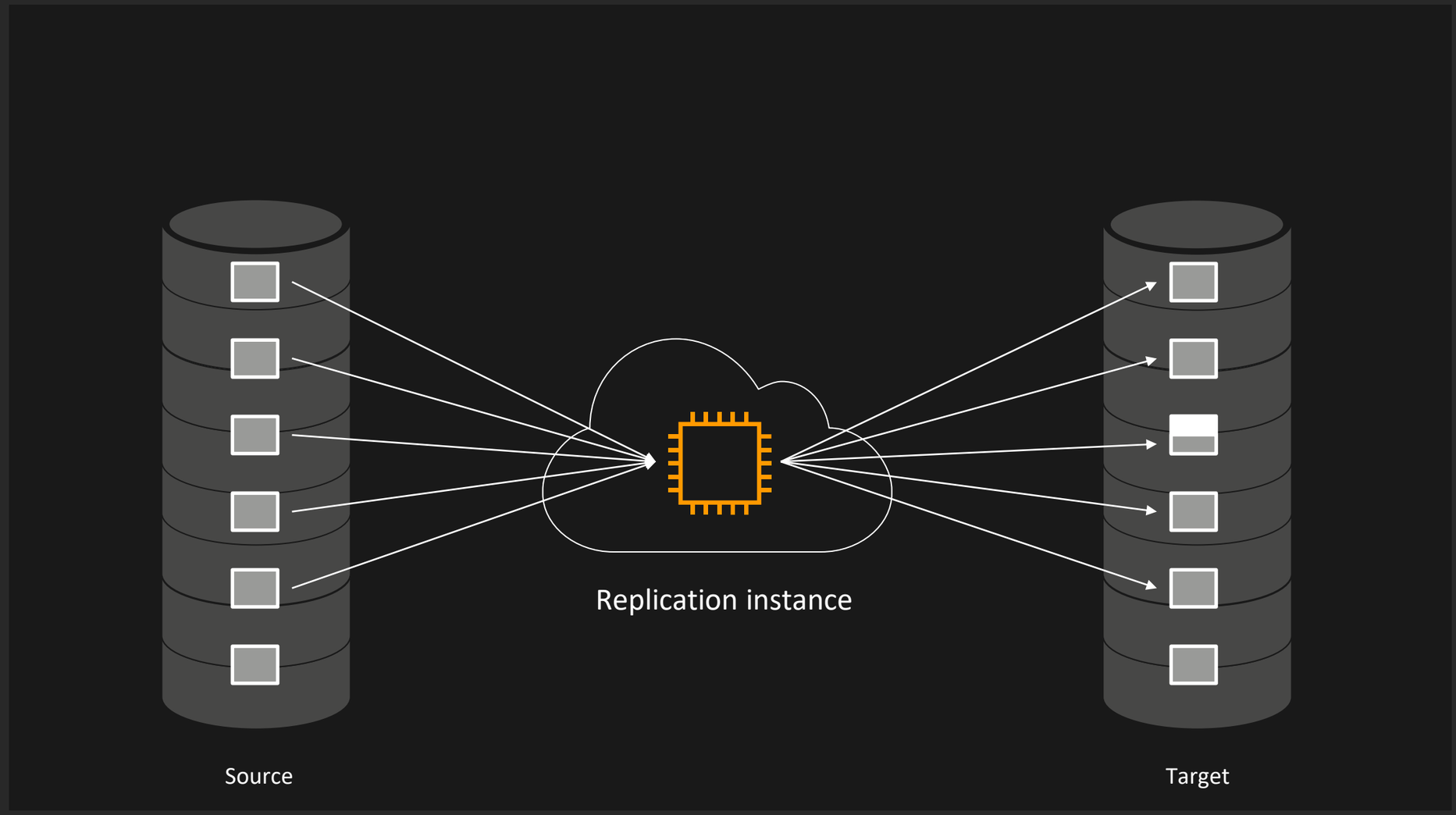
AWS DMS components—Tasks and replication instances

- Tasks run on a replication instances
- Contain two endpoints (source and target)
- Specify selection and transformation rules
- Replication instances support multiple tasks

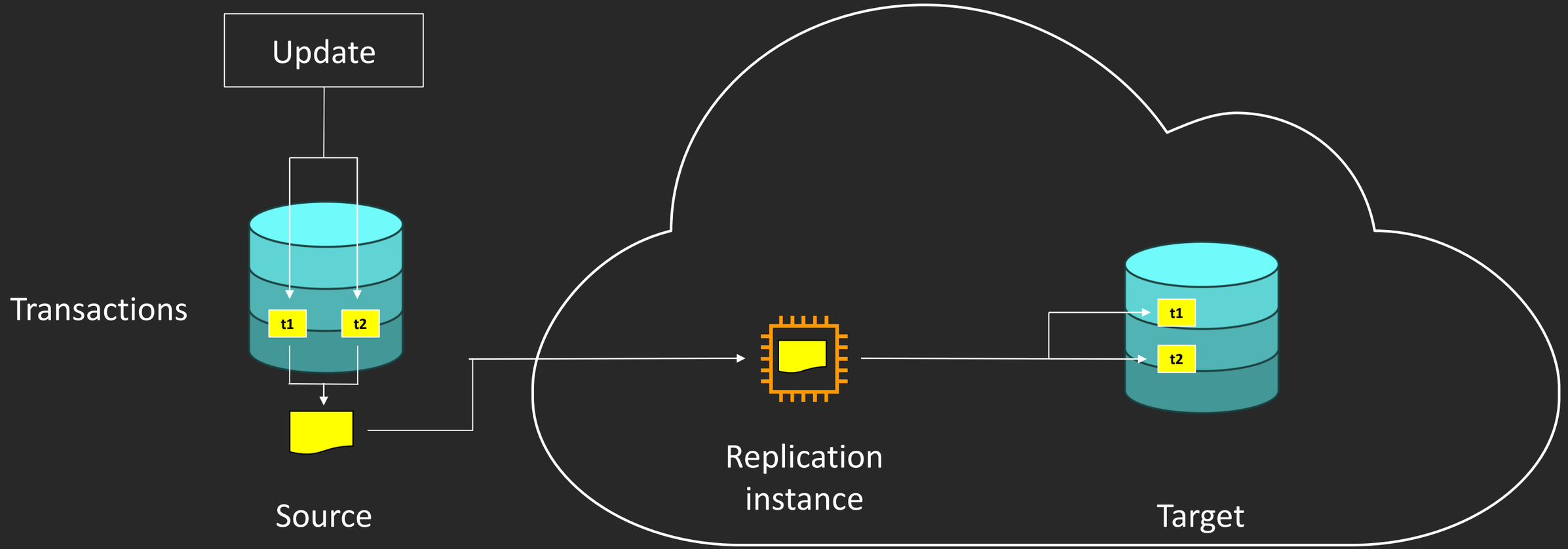
Rules and filters

- Selection rules—Applied at the source
  - Include or exclude schemas and tables
  - Filter rows based on column values
  - Can use multiple filters
- Transformation rules—Applied on the target
  - Rename schemas, tables, and columns
  - Remove columns, change case
  - Add/remove prefix or suffix

Migrate existing data (full load)



Change data capture (CDC)



Database migration procedure

1. Convert the schema using AWS SCT
2. Drop foreign keys and indexes on the target database
3. Set up a AWS DMS task to migrate initial data (full load)
4. Re-create foreign keys and indexes on the target database after the full load task is complete
5. Set up another AWS DMS task to capture and apply data changes (CDC) from the source
6. Point applications to the target database

aws

Migration best practices

Planning your migration

- Migration complexity analysis
  - Schema complexity analysis
  - Redo rate and TPS on source
  - Analyze the impact due to feature differences
  - Be aware of common migration issues
    (ex. https://wiki.postgresql.org/wiki/Oracle_to_Postgres_Conversion)

- Identify the pilot/quick win candidates

aws

Schema migration complexity analysis

Factor	Weightage	Comment
Tables	0.5	Non-partitioned tables will be converted automatically. Look for data type, character set issues.
Materialized views	3	Only fully refreshed Mviews are supported
Triggers	3	Review or rewrite the logic
PL/SQL procedures	4	Review or rewrite the logic
PL/SQL packages	15	Packages with >= 3 procedures or functions, review, or rewrite the logic
DB_link	1	Convert DBLinks using postgres_fdw extension
Partitions	0.1	Create all the child partitions manually. Apply the partitioning architecture with the help of functions/triggers.
LOB	1	LOB columns requires special handling

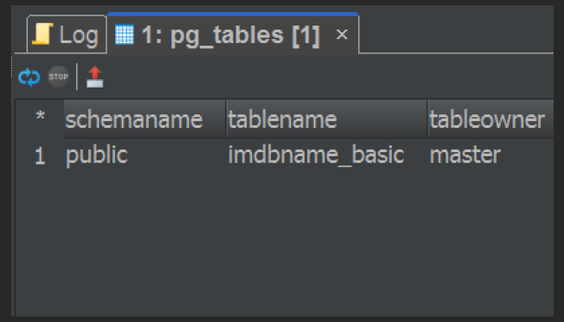
Schema migration complexity analysis—Example

DB_NAME	TAB	MV	DLINK	PRC	FUNC	PKG	TRG	Partitions	LOB	MAX_REDO_HR	TOT_WEIGHTAGE
DB1	'40:20'	'0:0'	'3:3'	'37:148'	'0:0'	'5:75'	'14:42'	‘50:5'	'0:0'	0.76	327

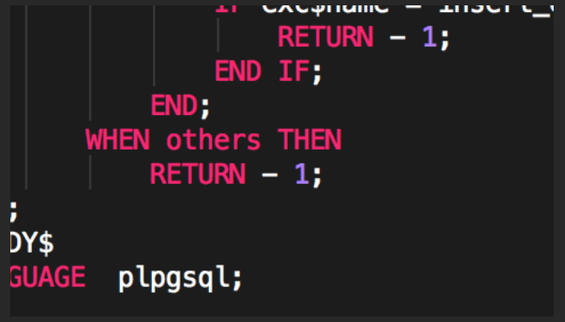
Factors	SMALL	MEDIUM	LARGE	XLARGE
Complexity	1–500	500–1000	1000–2000	2000+
Redo_GB_Per_Hour	0–4	4–10	10–20	20+

Number of DBs	SMALL	MEDIUM	LARGE	XLARGE
Org1	20	10	5	2
Org2	30	5	5	5

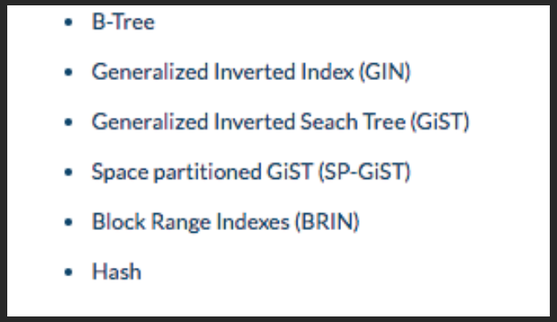
Understand basic database engine differences



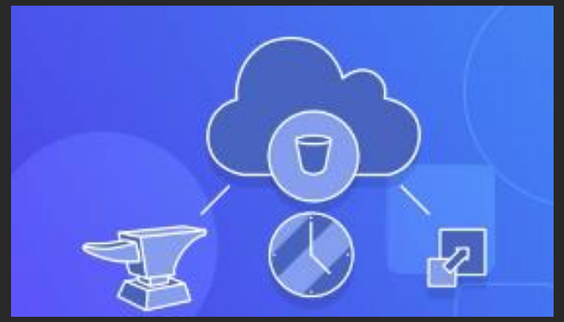
PostgreSQL is a lowercase data dictionary



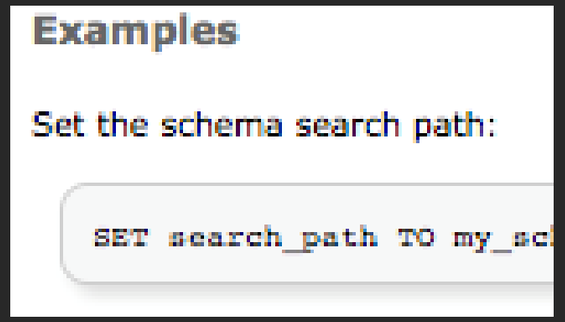
Use “exception handlers” when needed, not by default



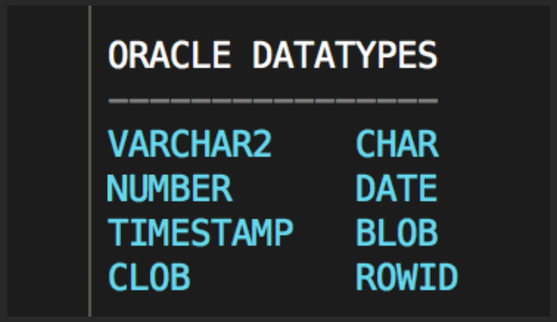
PostgreSQL has six index types



Store your BLOBs in Amazon S3 instead of the database

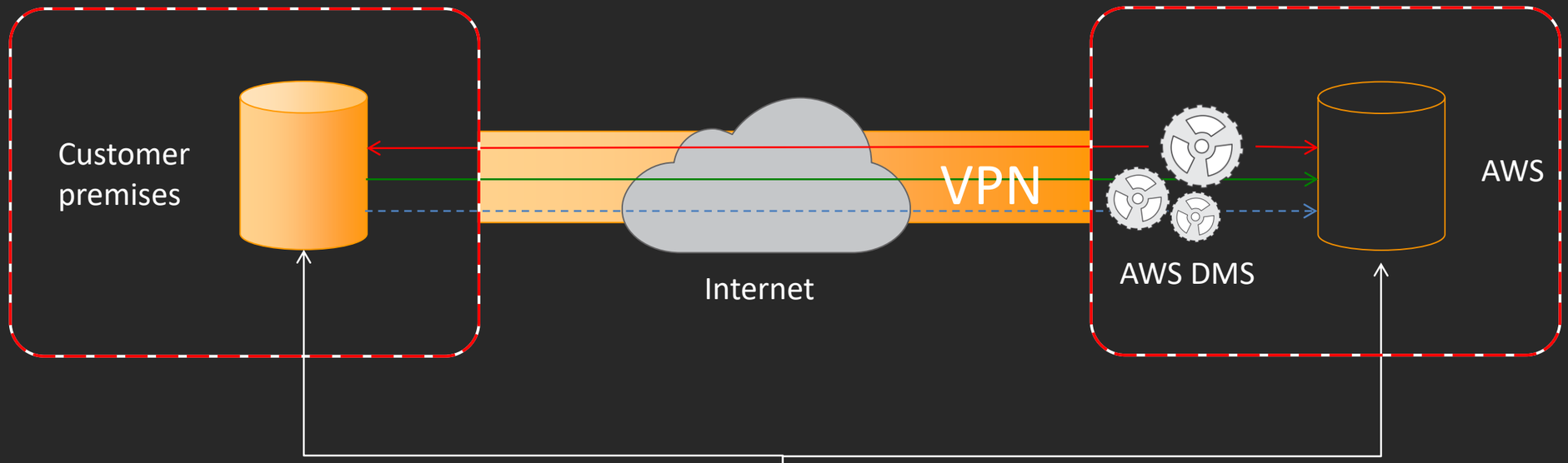


search_path replaces PUBLIC SYNONYM



PostgreSQL has 64 datatypes

Keep your apps running during migration



Start replication instance

Connect to source and destination

Select tables, schemas, or databases

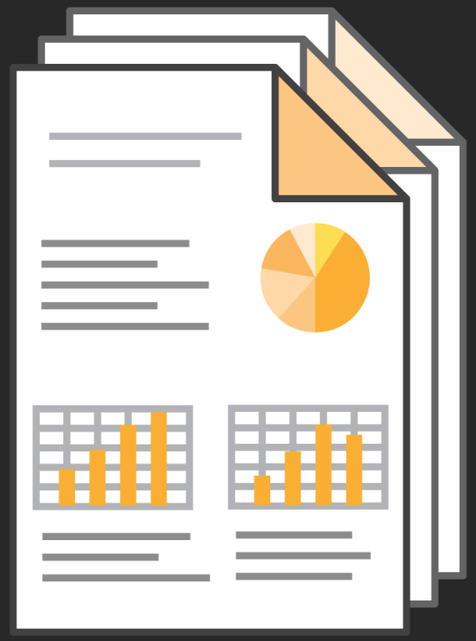
Application users

Let AWS DMS create tables, load data, and keep them in sync

Switch applications to new target at your convenience

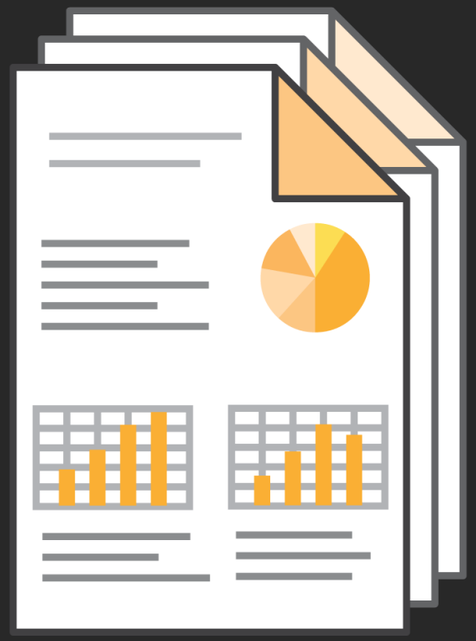
AWS DMS performance factors

- Resource availability on the source
- Network throughput
- Replication server capacity
- Number of objects, large tables
- LOB controls and performance



Large objects (LOBs) performance

- LOB controls
  - “Don’t include”—omit LOB columns
  - “Limited LOB mode”—specify “max LOB size”
  - “Full LOB mode”—specify “LOB chunk size”
- LOB performance
  - Whenever possible use “limited LOB mode”



Performance with high TPS source systems

Transactional apply (default)

Maintains transaction commit order

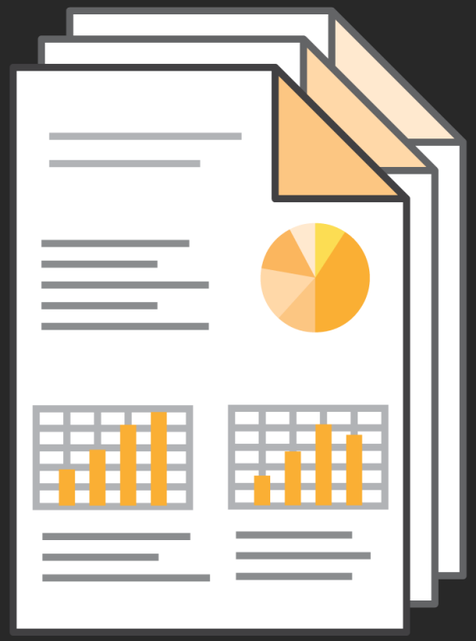
Batch apply

Build a batch with changes from source

Intermediate net changes on target

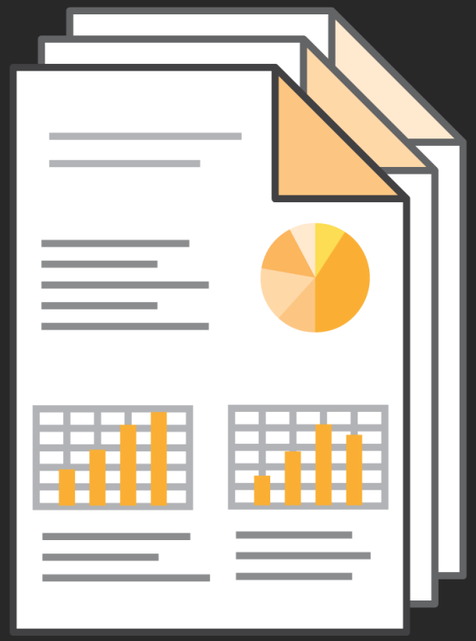
Condense changes and apply

- Applies all inserts, all updates, and all deletes in the same order

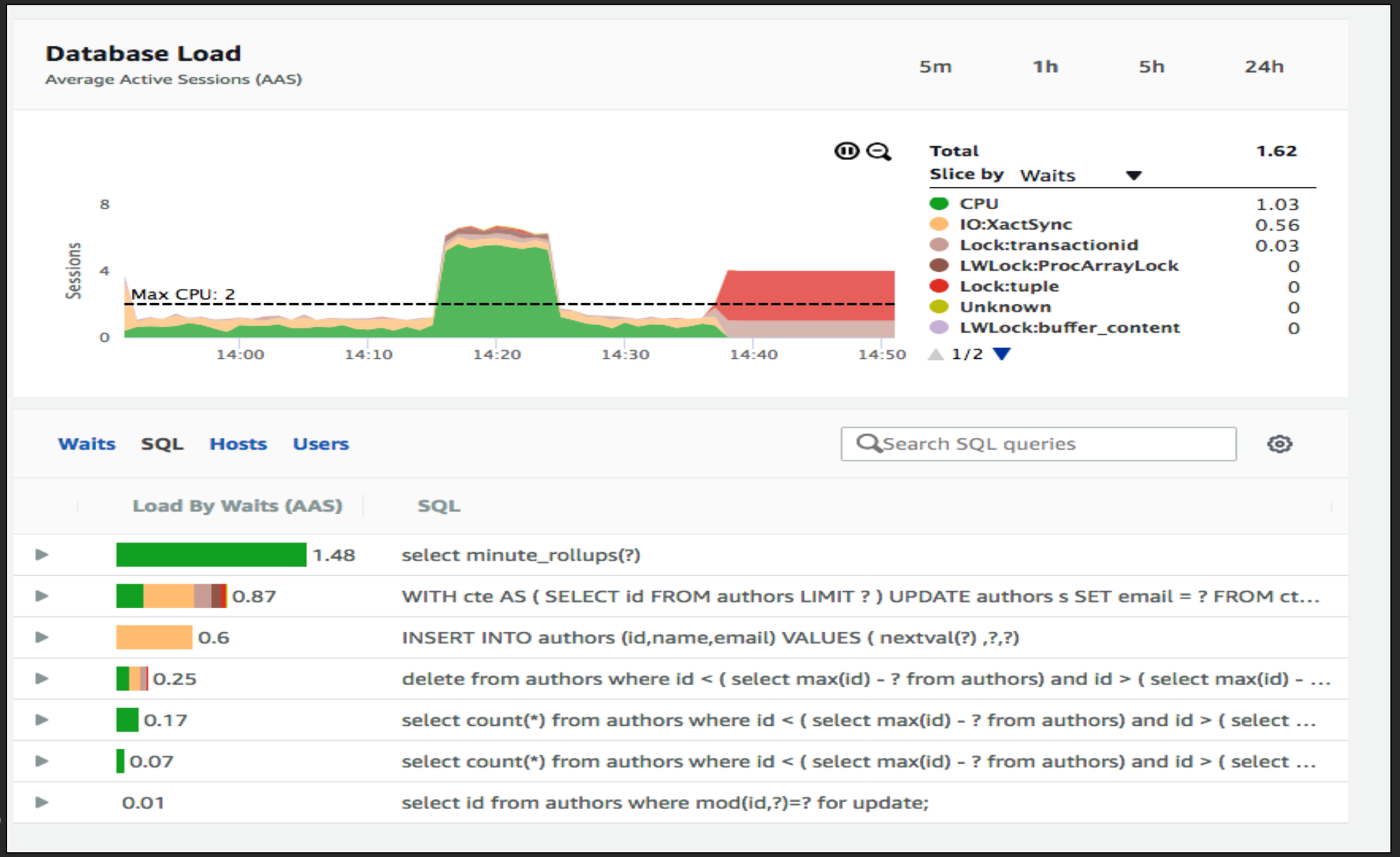


Performance testing

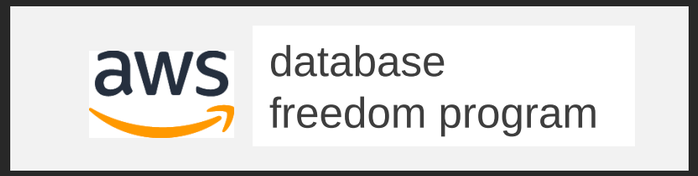
- Load testing
- Identify critical application features
- Features with known performance issues
- Identify Top SQLs/complex SQLs



Performance insights



How AWS can help



Tools

Programs

Partners

aws

AWS database migration partners

Thank You