Biggest migrations from Oracle to PostgreSQL

With 2000 functions and 1.5 TB data

Viswanatha Shastry M
Viswa.medipalli@openscg.com

OpenSCG

BIGSQL

About OpenSCG

- Started Operations in early 2010
- Around 130 member team
- Headquartered in East Brunswick, NJ
- Presence in
  - Rochester, NY
  - San Mateo, CA
  - Hyderabad, India
  - Bangalore, India

PostgreSQL Experts

Long Community Involvement & Leadership

AWS Consulting Partner

OpenSCG

BIGSQL

Agenda

- Schema Migration - Data Type Mapping
- Schema Migration - Object Mapping
- Views
- Functions
- Data Migration
- Unit Testing
- Execution
- Work Around

Schema Migration – Data Type Mapping

- Why Data types are Important?

Schema Migration – Data Type Mapping

- Oracle Data types
  - Number
  - Varchar2
  - Date
- Oracle Large Data types
  - CLOB
  - BLOB

Schema Migration – Data Type Mapping

- Numeric data types
  - PostgreSQL provides six types of numeric data types
  - Four are exact value (SMALLINT, INTEGER, BIGINT, NUMERIC(p,s))
    - Three of them store only integers and fourth (NUMERIC(p,s)) can accurately store any value that fits within the specified number (p) of digits.
  - Two are approximate (REAL, DOUBLE PRECISION).

Schema Migration – Data Type Mapping – Numbers

- Numbers → SmallInt (2B)
  → Integer (2B)
  → BinInt (4B)
  → Double Precision (8B)
  → Float (4B)
  → Numeric/Decimal Variable

OpenSCG BIGSQL

Schema Migration – Data Type Mapping –Varchars

- Clob → Text
- Varchar2 → Text
  → Character Varying
  → Varchar
  → Varchar(n)
  → Char(n)
- Date → Date
  → Timestamp

Schema Migration

- Data type mapping with the tool

OpenSCG - Schema Migration App

Oracle to PostgreSQL data type mapping

	Oracle		PostgreSQL
1	BFILE	Pointer to binary file, ⇐ 4G	VARCHAR(255)
2	BINARY_FLOAT	32-bit floating-point number	REAL
3	BINARY_DOUBLE	64-bit floating-point number	DOUBLE PRECISION
4	BLOB	Binary large object, ⇐ 4G	BYTEA
5	CHAR(*n*), CHARACTER(*n*)	Fixed-length string, 1 ⇐ *n* ⇐ 2000	CHAR(*n*), CHARACTER(*n*)
6	CLOB	Character large object, ⇐ 4G	TEXT
7	DATE	Date and time	TIMESTAMP(0)
8	DECIMAL(*p*,*s*), DEC(*p*,*s*)	Fixed-point number	DECIMAL(*p*,*s*), DEC(*p*,*s*)
9	DOUBLE PRECISION	Floating-point number	DOUBLE PRECISION
10	FLOAT(*p*)	Floating-point number	DOUBLE PRECISION
11	INTEGER, INT	38 digits integer	DECIMAL(38)
12	INTERVAL YEAR(*p*) TO MONTH	Date interval	INTERVAL YEAR TO MONTH
13	INTERVAL DAY(*p*) TO SECOND(*s*)	Day and time interval	INTERVAL DAY TO SECOND(*s*)
14	LONG	Character data, ⇐ 2G	TEXT
15	LONG RAW	Binary data, ⇐ 2G	BYTEA
16	NCHAR(*n*)	Fixed-length UTF-8 string, 1 ⇐ *n* ⇐ 2000	CHAR(*n*)
17	NCHAR VARYING(*n*)	Varying-length UTF-8 string, 1 ⇐ *n* ⇐ 4000	VARCHAR(*n*)
18	NCLOB	Variable-length Unicode string, ⇐ 4G	TEXT

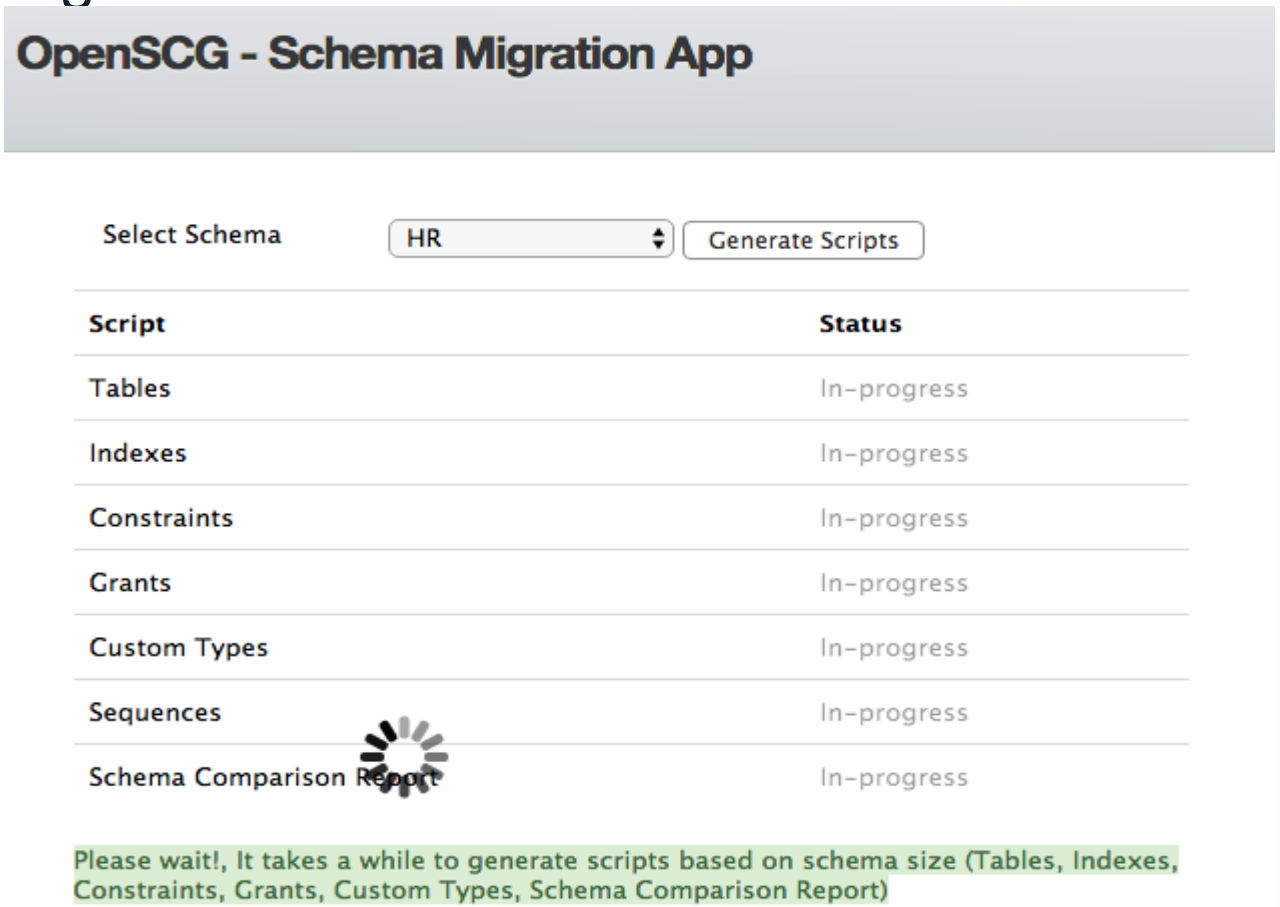
OpenSCG

BIGSQL

Schema Migration

- Create SQL files for each of the object types
  - Tables
  - Indexes
  - Comments
  - Constraints
  - Custom data types
  - Sequences
  - Comparison report – CSV file

Schema Migration

- Generating the files
  - 

OpenSCG

BIGSQL

Schema Migration

- Comparison Report

Table Name	Columns name	Data_type	Data_length	Data_precision	Nullable	Default	Table Name	Columns	Data_type	Data_length	Data_precision	Nullable	Table Name	Columns nam	Data_type	Data_length	Data_precisio	Nullable	Default
ACCOUNT	ACCOUNTID	NUMBER	22		N		Y	Y	Y	N	N	Y	account	accountid	bigint	64	2	NO	
ACCOUNT	ACCOUNTNA	VARCHAR2	150		Y		Y	Y	Y	Y	Y	Y	account	accountname	character vary	150		YES	
ACCOUNT	BROKERACC	VARCHAR2	200		Y		Y	Y	Y	Y	Y	Y	account	brokeraccount	character vary	200		YES	
ACCOUNT	BROKERID	NUMBER	22		Y		Y	Y	Y	N	N	Y	account	brokerid	integer	32	2	YES	
ACCOUNT	CLIENTID	NUMBER	22		Y		Y	Y	Y	N	N	Y	account	clientid	integer	32	2	YES	
ACCOUNT	CREATE_DAT	DATE	7		Y		Y	Y	Y	N	Y	Y	account	create_date	date			YES	
ACCOUNT	MARGIN_CCY	NUMBER	22		Y		Y	Y	Y	N	N	Y	account	margin_ccytyp	smallint	16	2	YES	
ACCOUNT	MARGIN_ROL	DATE	7		Y		Y	Y	Y	N	Y	Y	account	margin_rolling	date			YES	
ACCOUNT	MIN_TRANSF	NUMBER	22		Y		Y	Y	Y	N	N	Y	account	min_transfer_	numeric		10	YES	
ACCOUNT	TRANSFER_F	NUMBER	22		Y		Y	Y	Y	N	N	Y	account	transfer_round	integer	32	2	YES	
ACCOUNT_DA	ACCOUNT_DA	NUMBER	22		N		Y	Y	Y	N	N	Y	account_data	account_data_	bigint	64	2	NO	
ACCOUNT_DA	BALANCE	NUMBER	22		N		Y	Y	Y	N	N	Y	account_data	balance	numeric		10	NO	

OpenSCG

BIGSQL

Object Mapping -- Oracle to PstgreSQL

- Tables → Tables
- Synonyms → Views / set search-path
- Packages → Schema's
- Procedures → Functions
- Functions → Functions

Views

- Try Avoiding with views
  - Too many function calls
  - Predicates will not get pushed if have window functions
  - Will query all columns irrespective of columns selected from view
  - Nested views

Functions

- Immutable
  - Indicates that the function cannot modify the database and always returns the same result when given the same argument values
  - Optimizer to pre-evaluate the function when a query calls it with constant arguments

Functions

- Stable
  - Indicates that the function cannot modify the database, and that within a single table scan it will consistently return the same result for the same argument values
  - Will be appropriate selection for functions whose results depend on database lookups, parameter variables
  - current_timestamp family of functions qualify as stable, since their values do not change within a transaction.

Functions

- Volatile
  - Indicates that the function value can change even within a single table scan, so no optimizations can be made.
  - An example is random().

Data Migration

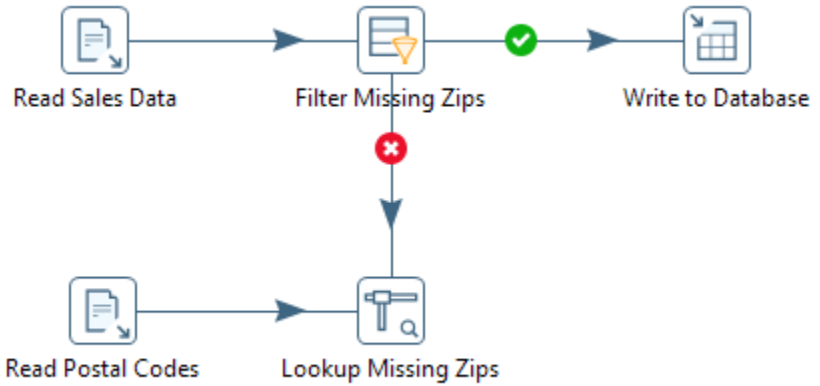
- How to migrate 1.5TB of data with replication on 5 servers and located in 2 different regions with least possible downtime?

Data Migration

- Before data migration make sure
  - No Indexes
  - No Constraints
  - Plain Tables with comments

Data Migration – Pentaho ETL

- ETL is a process for pulling data out of the source systems and placing it into target systems.
- Graphical extract-transform-load (ETL) designer to simplify the creation of data pipelines; Rich library of pre-built components to access, prepare, and blend data from relational sources



OpenSCG

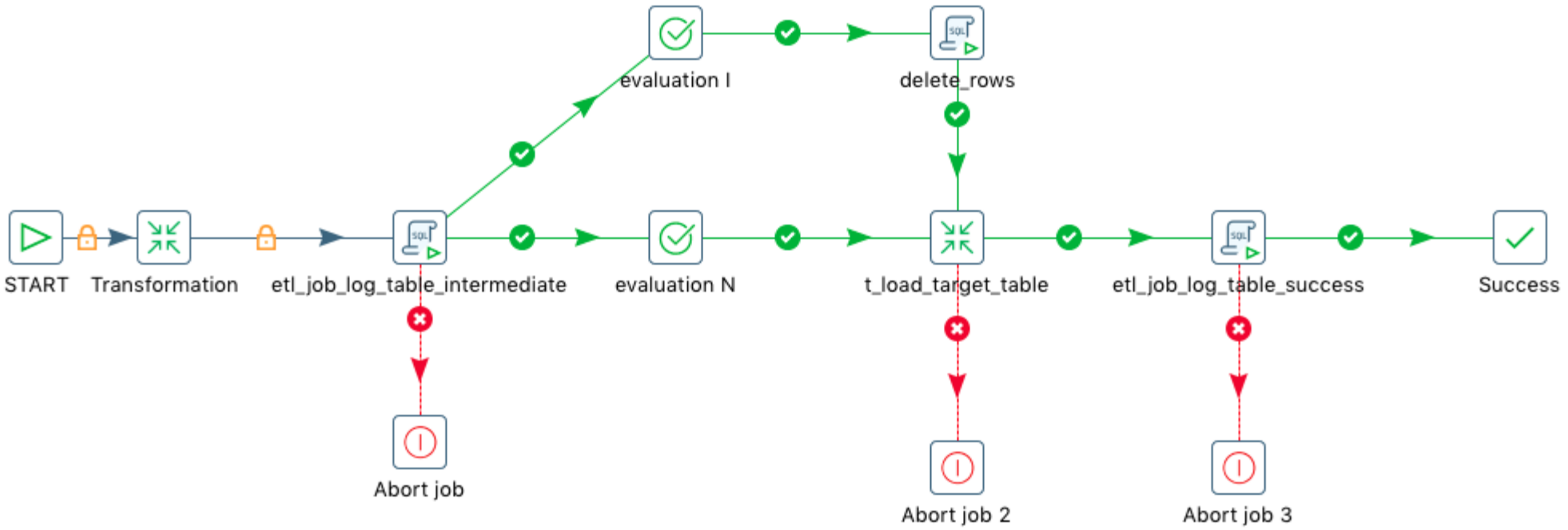
BIGSQL

Data Migration – Pentaho ETL

- Data identification to migrate in multiple chunks
- Configuration table
  - ```
    CREATE TABLE  schema.etl_job_log_table(seq integer,
    schema_name character varying(30),table_name character
    varying(30), select_clause_cols character
    varying(4000),  column_name character varying(200),
    column_value character varying(300),operator character
    varying(500),  column_list character varying(4000),
    source_record_cnt integer,  target_record_cnt integer,
    status character(1),  batch_id integer,start_ts
    timestamp(6) without time zone,  end_ts timestamp(6)
    without time zone,);
    ```
    ```

Data Migration – Pentaho ETL

- Dynamic table mapping



OpenSCG

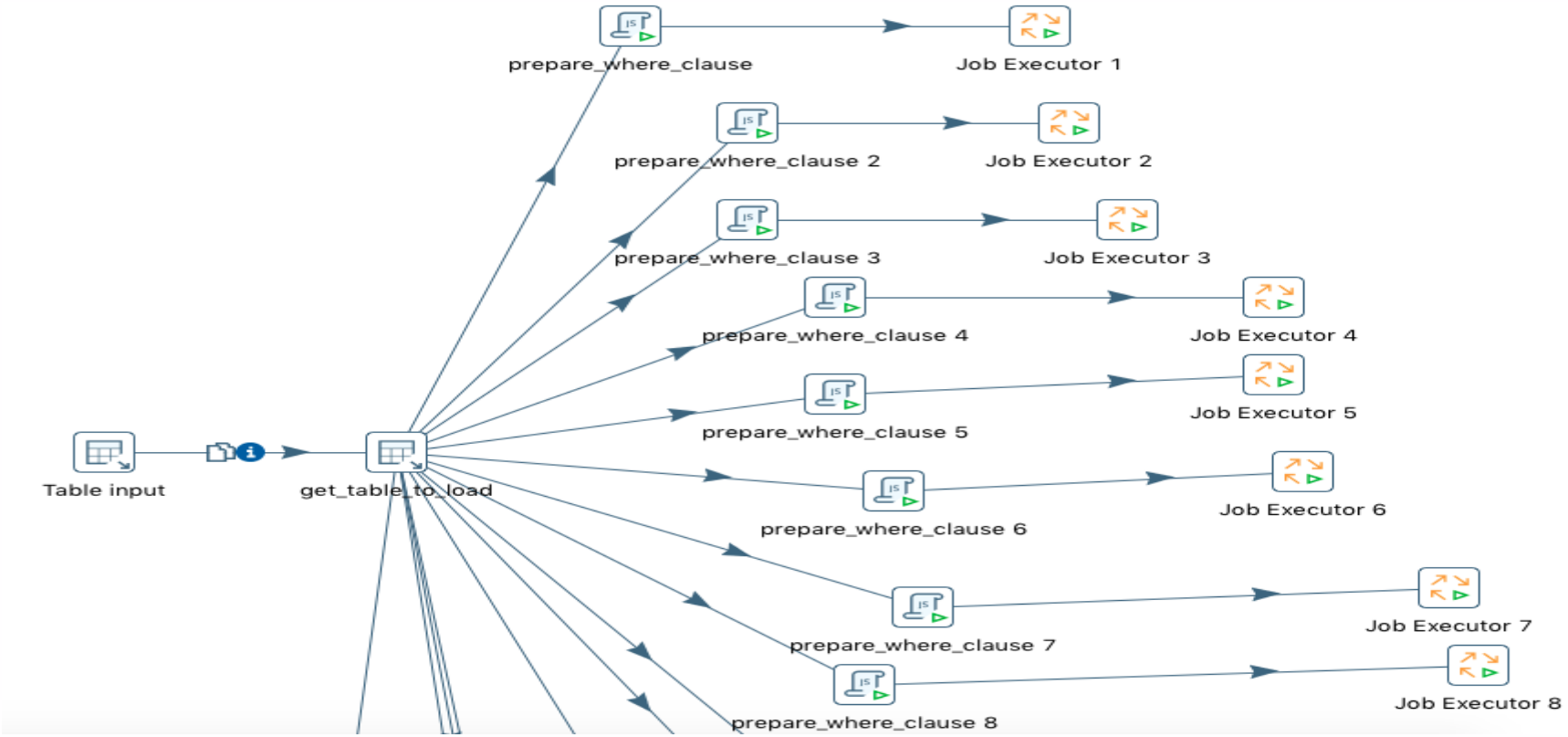
BIGSQL

Data Migration – Pentaho ETL

- Splitting tables
  - Select_clause_cols column from configuration table will provide what columns need to be pushed to target table.
  - Column_list will provide the order of columns in the target system

Data Migration – Pentaho ETL

- Multiple chunks running at the same times



- Slower as they use “Insert into”

OpenSCG

BIGSQL

Data Migration – FDW

- FDW or foreign-data wrapper can be used to access data stored in external sources
- This functionality of the older dblink.
- Create a foreign server object , user mapping and foreign table to access the data from external source
- Refer URL for how to create Server and other objects
- https://www.postgresql.org/docs/9.5/static/postgres-fdw.html

Data Migration – FDW

- Data identification to migrate in multiple chunks
- Configuration table
  - `CREATE TABLE schema.etl_job_log_table(seq integer, schema_name character varying(30), table_name character varying(30), select_clause_cols character varying(4000), column_name character varying(200), column_value character varying(300),operator character varying(500), column_list character varying(4000), source_record_cnt integer, target_record_cnt integer, status character(1), batch_id integer,start_ts timestamp(6) without time zone, end_ts timestamp(6) without time zone,);`

Data Migration – FDW

- SQL Files
  - SQL for file set the status for in-progress

```
Do $$
Begin
   Update etl_job_log_table set status='I'
   End $$
```

  - SQL file with anonymous code block for data migration and set the status to completed

```
Do $$
Begin
     Insert into table1(col1,col2,col3) Select col1,col2,col3 from
   foreign_schema.foreign_table
         Update etl_job_log_table set status='Y'
End $$
```

Data Migration – FDW

- Splitting tables
- Pre-Fetch 10k records
- Multiple chunks running at the same times
- Slower as they use “Insert into”

Data Migration – ora2pg

- Its free tool used to migrate an oracle to PostgreSQL compatible schema.
- Has features to migrate
  - Tables
  - Indexes
  - Constraints
  - Views
  - Functions
  - Data migration

Data Migration – ora2pg

- Configurations
  - For Source and target systems
    - ORACLE_DSN,ORACLE_USER, ORACLE_PWD
    - PG_DSN,PG_USER, PG_PWD
  - For list of tables to migrate
    - ALLOW
  - For data migration limit by default 1m
    - DATA_LIMIT
  - For data migration with views
    - REPLACE_QUERY
      - TableY [SELECT * FROM TableY WHERE AS_OF_DT < 20100101]

Data Migration – ora2pg

- Data identification with multiple chunks
- Multiple configuration files
- Splitting tables
- Multiple chunks running at the same times

Data Migration – ora2pg

- Performance

Schema - Table	No of Fields	Partition	Tool	Environment	No of recs	Threads	Data Limit	CPU	recs / sec
Table1	6	No	Ora2pg	UAT	25605580	4	1 million	60 - 95%	23494
Table2	14	No	Ora2pg	UAT	22646665	4	1 million	60 - 95%	7772
Table3	4	No	Ora2pg	UAT	21943125	4	1 million	60 - 95%	901
Table4	21	No	Ora2pg	UAT	21209051	4	1 million	60 - 95%	5434
Table6	21	No	Ora2pg	UAT	65471220	7	1 million	90-100%	5725
Table7	16	No	Ora2pg	UAT	**63475890**	7	1 million	90-100%	6990
Table8	19	No	Ora2pg	UAT	63347103	7	1 million	90-100%	5981
Table9	33	No	Ora2pg	UAT	**62133303**	7	1 million	90-100%	3034
Table10	2	No	Ora2pg	UAT	56119278	7	2 million	90-100%	18687
Table11	26	No	Ora2pg	UAT	49168738	7	3 million	90-100%	4460
Table12	6	No	Ora2pg	UAT	**38968098**	7	4 million	90-100%	8024

OpenSCG

BIGSQL

Data Migration – Data Validation/Verification

- Compare the data by
  - Aggregate functions against Numeric, Varchar and Date columns
  - Extract the data from Views for key data
  - Extract the data from function returned values

Unit testing

- Compare the data by
  - Executing all the functions
  - Data consistency before and after function execution of functions
  - Complete code coverage

Unit testing

- Compare the data by
  - Performance of functions comparison with source and target with execution timings

	A	B	C	D	E	F	G	H	I	J	
1	Package Name	TestCase Name	Block Name	Query	Source Statu	Target S	Source N	Target No.	Source Execution	Target Execution Tim	Errors
2	PKG_sample_test	p_sample_1.xml	before	Query-1	Pass	Pass	2	2	534	1534	
3	PKG_sample_test	p_sample_1.xml	actual	Query-1	Pass	Pass	0	0	357	0	
4	PKG_sample_test	p_sample_1.xml	after	Query-1	Pass	Pass	3	3	499	509	
5											
6	PKG_sample_test	p_sample_2.xml	before	Query-1	Pass	Pass	2	0	508	494	
7	PKG_sample_test	p_sample_2.xml	actual	Query-1	Pass	Pass	0	0	305	0	
8	PKG_sample_test	p_sample_2.xml	after	Query-1	Pass	Pass	3	1	485	479	
9											
10	PKG_sample_test	p_sample_3.xml	before	Query-1	Pass	Pass	2	2	488	503	
11	PKG_sample_test	p_sample_3.xml	actual	Query-1	Pass	Pass	0	0	315	0	
12	PKG_sample_test	p_sample_3.xml	after	Query-1	Pass	Pass	2	2	494	486	
13											
14	PKG_sample_test	p_sample_4.xml	before	Query-1	Pass	Pass	1	5	509	489	
15	PKG_sample_test	p_sample_4.xml	actual	Query-1	Pass	Pass	0	0	294	0	
16	PKG_sample_test	p_sample_4.xml	after	Query-1	Pass	Pass	1	5	510	449	
17											
18	PKG_sample_test	p_sample_5.xml	before	Query-1	Pass	Pass	10	1	515	442	
19	PKG_sample_test	p_sample_5.xml	actual	Query-1	Pass	Pass	0	0	11920	0	
20	PKG_sample_test	p_sample_5.xml	after	Query-1	Pass	Pass	10	1	513	474	
21											
22	PKG_sample_test	p_sample_6.xml	before	Query-1	Pass	Pass	1	1	511	496	
23	PKG_sample_test	p_sample_6.xml	actual	Query-1	Pass	Pass	0	0	355	0	
24	PKG_sample_test	p_sample_6.xml	after	Query-1	Pass	Pass	1	1	535	739	

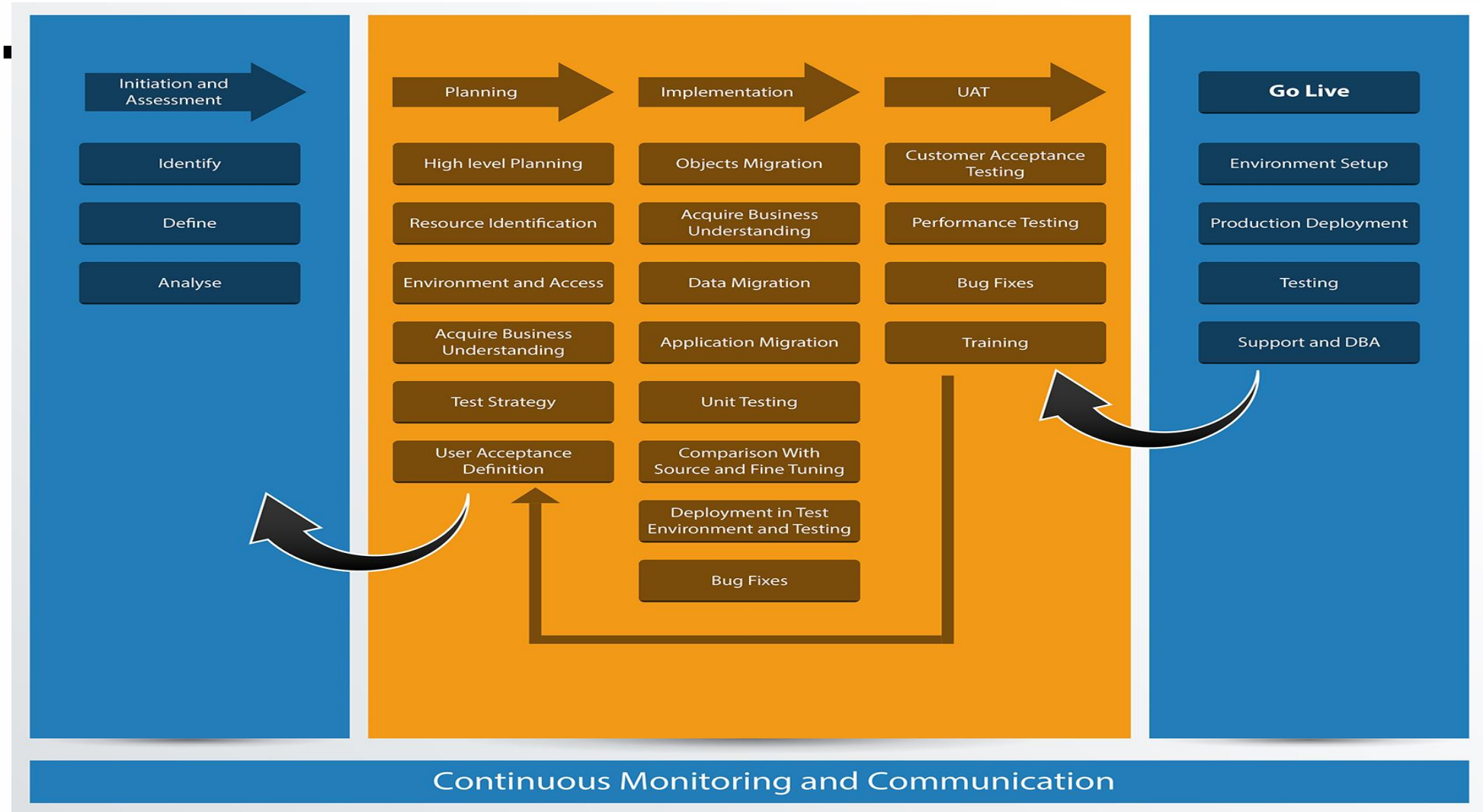
OpenSCG

BIGSQL

Execution

- Identify independent modules and database object dependencies
- Plan for multiple interim deliverable
- Incremental testing
- Cut-over Plan
  - Pre-Migration
  - Migration
  - Post-Migration

Execution



OpenSCG

BIGSQL

Work Around

- LISTAGG - STRING_AGG
- SEQUENCE.NEXTVAL – NEXTVAL('SEQUENCE')
- DBMS_APPLICATION_INFO – APPLICATION_NAME
- DBMS_CRYPTO – PG_CRYPTO

Work Around

- MERGE – CTE
- NVL – COALESCE
- DISTINCT WITH ORDER BY – CTE
- DBMS_OUTPUT.PUT_LINE – RAISE INFO

Questions

- Q&A

Viswanatha Shastry M
Viswa.medipalli@openscg.com

- Thank you