aws professional services

PostgreSQL for Oracle Users

March 2021

DUAL

ORACLE

```
SELECT sysdate FROM dual;
```

Postgres

```
Select now(); -- transaction
select clock_timestamp; -- increment

create table dual as (values(true));
```

GIVE ME 10 rows

ORACLE

```
SELECT rownum
FROM dual
CONNECT BY rownum <= 10;
```

Postgres

```
SELECT generate_series
FROM generate_series(1, 10);

Or
SELECT ..From a table
Limit 10;
```

At least math is easy

aws professional services

Integer arithmetic

ORACLE

```
SELECT 1/4 quarter_pounder
FROM dual;
```

PostgreSQL

```
SELECT 1/4 royale_with_cheese;
```

This is madness!

This is ***integer arithmetic!***

```
10/4 = 2
5+10/4 = 7
5.0+10/4 = 7.0
SELECT avg(x) FROM foo;
      0
```

```
10/4.0 = 2.5
5+10/4.0 = 7.5
10/4::float = 2.5
SELECT avg(x::float) FROM foo;
    0.22
```

It's a date

Date and time types

ORACLE

DATE – Date+time with second resolution
TIMESTAMP – Date+time with microsecond resolution

- Optionally can encode time zone

Postgres

DATE – Date with one day resolution
TIMESTAMP – Date + time with microsecond resolution

CREATE a DATE

ORACLE

```
SELECT to_date('2016/03/26')
FROM dual;

2016/03/26
```

You **should** specify a date format.

Postgres

```
SELECT to_date('2016/03/26');

Invalid operation: function
to_date("unknown") does not
exist;
```

You **must** specify a date format.

Date math

ORACLE

```
date_col1 - date_col2 = number
```

```
SS
```

Postgres

```
date_col1 - date_col2 = integer

tstamp_col1 - tstamp_col2 = interval

date_col1 - tstamp_col2 = interval

tstamp_col1 - date_col1 = interval
```

Date math

Oracle

```
date_col + 1 = 2018/04/04
tstamp_col + 1 = 2018/04/04
date_col + 0.5 = 2018/04/04 12:00
```

Postgres

```
date_col + 1 = 2018/04/04
tstamp_col + '1 day'::interval = 2018/04/04
date_col + '0.5'::interval = "2018/04/040.5"
```


aws professional services

More date fun!

TRUNC(timestamp) returns a DATE…

- ..but only a date by truncating time portion to a date

DATE_TRUNC() works more like Oracle trunc()

- …but the parameters are "backward" and
- …there are a whole new set of datepart abbreviations

date_trunc('mon', sysdate) = 2016-03-01 00:00:00

- …and….

Weeks start on Monday

Oracle

```
trunc(t, 'day') = Sunday
```

Postgres

```
date_trunc('week', d)) = Monday
date_trunc('week', d+1)-1 = Sunday
date_trunc('week', d+1)-1+6 = Saturday
```

```
select date_trunc('week', now() + '1 day'::interval)-'1
day'::interval
```

Looking at strings

NULL ”Handling is Different"

```
1 + NULL = NULL
sysdate + NULL = NULL
SUM(all_null_column) = NULL
SUM(some_nulls_column) → Ignore Nulls
```

- Oracle

'string' || NULL = 'string'

- PostgreSQL

'string' || NULL = NULL

Searching strings

LIKE exists but so does ILIKE

- lower(foo) LIKE '%interesting%'
- foo ILIKE '%interesting%'

Regex is done by it's own operator

- lower(foo) ~ '.*interesting.*'

Use LIKE/ILIKE if you can

Postgres DDL Standards

aws professional services

Oracle

```
CREATE TABLE booker.ad_decorations
( advertiser_idNUMBER(38)
, ad_decoration_idNUMBER(38)NOT NULL
, ad_group_idNUMBER(38)NOT NULL
, ad_idNUMBER(38)NOT NULL
, created_byVARCHAR2(8)NOT NULL
, creation_dateDATENOT NULL
, decoration_idNUMBER(38)NOT NULL
, decoration_natural_idVARCHAR2(255)NOT NULL
, decoration_orderNUMBER(10)NOT NULL
, internal_statusNUMBER(10)NOT NULL
, last_updated_byVARCHAR2(8)NOT NULL
, last_updated_dateDATENOT NULL
, marketplace_idNUMBER(38)
, statusNUMBER(10)NOT NULL
, uniqueness_scopeVARCHAR2(64)NOT NULL
, versionNUMBER(38)NOT NULL
)
PCTFREE 10
PCTUSED
INITRANS 8
TABLESPACE bid
STORAGE ( MAXEXTENTS UNLIMITEDPCTINCREASE 0FREELISTS)
/
```


Postgres

```
CREATE TABLE booker.ad_decorations(
ad_decoration_id NUMERIC(38,0) NOT NULL,
ad_id NUMERIC(38,0) NOT NULL,
decoration_id NUMERIC(38,0) NOT NULL,
marketplace_id NUMERIC(38,0),
advertiser_id NUMERIC(38,0),
decoration_order NUMERIC(10,0) NOT NULL,
status NUMERIC(10,0) NOT NULL,
ad_group_id NUMERIC(38,0) NOT NULL,
decoration_natural_id CHARACTER VARYING(255) NOT
NULL,
uniqueness_scope CHARACTER VARYING(64) NOT NULL,
internal_status NUMERIC(10,0) NOT NULL,
version NUMERIC(38,0) NOT NULL,
created_by CHARACTER VARYING(64) NOT NULL,
creation_date TIMESTAMP(0) WITH TIME ZONE NOT
NULL default (clock_timestamp() at time zone
'UTC'),
last_updated_by CHARACTER VARYING(64) NOT NULL,
last_updated_date TIMESTAMP(0) WITH TIME ZONE
NOT NULL,
program_type NUMERIC(5,0)
)
WITH (
OIDS=FALSE
);
```

aws professional services

```
CREATE INDEX booker.pk_ad_decorations
ON booker.ad_decorations
( ad_decoration_id )
PCTFREE 10
INITRANS 8
TABLESPACE bid_idx
STORAGE ( MAXEXTENTS UNLIMITEDPCTINCREASE 0 )
ONLINE;
ALTER TABLE booker.ad_decorations
ADD CONSTRAINT pk_ad_decorations
PRIMARY KEY ( ad_decoration_id )
USING INDEX;
```

```
ALTER TABLE booker.ad_decorations
ADD CONSTRAINT pk_ad_decorations
PRIMARY KEY (ad_decoration_id);
```

```
CREATE INDEX booker.uq_ads_decors_uq_scope
ON booker.ad_decorations
( ad_group_id, uniqueness_scope, decoration_id )
PCTFREE 10
INITRANS 8
TABLESPACE bid_idx
STORAGE ( MAXEXTENTS UNLIMITEDPCTINCREASE 0 )
ONLINE;
ALTER TABLE booker.ad_decorations
ADD CONSTRAINT uq_ads_decors_uq_scope
UNIQUE ( decoration_id, uniqueness_scope,
ad_group_id )
USING INDEX;
```

```
ALTER TABLE booker.ad_decorations
ADD CONSTRAINT uq_ads_decors_uq_scope
UNIQUE (ad_group_id, uniqueness_scope,
decoration_id );
```

```
CREATE OR REPLACE TRIGGER booker.AUDIT_AD_DECORATIONS
BEFORE INSERT OR UPDATE
ON booker.ad_decorations
FOR EACH ROW
DECLARE
osuserVARCHAR2(8 BYTE);
BEGIN
--
-- get who is making the DML change from the session
--
osuser := session_info.osuser;
--
IF (NOT (dbms_snapshot.i_am_a_refresh)
AND dbms_reputil.replication_is_on) THEN
IF NOT dbms_reputil.from_remote THEN
IF INSERTING THEN
:new.created_by := osuser;
:new.creation_date := sysdate;
:new.last_updated_by := osuser;
:new.last_updated_date := sysdate;
END IF;
--
IF UPDATING THEN
:new.last_updated_by := osuser;
:new.last_updated_date := sysdate;
END IF;
END IF;
END IF;

EXCEPTION

WHEN OTHERS THEN
-- raise error if we cannot set auditing fields
raise_application_error(-20505, 'BOOKER.AUDIT_AD_DECORATIONS
Failed. Rolling Back.' || SQLERRM);
END;
/
```



```
CREATE OR REPLACE FUNCTION booker.audit_trg_function()
 RETURNS trigger AS
$BODY$
BEGIN
       IF (TG_OP = 'INSERT') THEN
              NEW.creation_date=clock_timestamp() AT TIME ZONE 'UTC';
              NEW.last_updated_date=clock_timestamp() AT TIME ZONE 'UTC';
              NEW.created_by=SESSION_USER::TEXT;
              NEW.last_updated_by=SESSION_USER::TEXT;
              RETURN NEW;
       ELSIF (TG_OP = 'UPDATE') THEN
              NEW.last_updated_date=clock_timestamp() AT TIME ZONE 'UTC';
              NEW.last_updated_by=SESSION_USER::TEXT;
              RETURN NEW;
       ELSE
        RAISE WARNING 'audit_trg_function exeception in %.% - OTHER ACTION
OCCURRED: %, AT %',TG_TABLE_SCHEMA,TG_TABLE_NAME,TG_OP,clock_timestamp();
        RETURN NULL;
       END IF;
EXCEPTION
    WHEN DATA_EXCEPTION THEN
        RAISE WARNING 'audit_trg_function exeception in %.% - UDF ERROR [DATA
EXCEPTION\] - SQLSTATE: %, SQLERRM:
%',TG_TABLE_SCHEMA,TG_TABLE_NAME,SQLSTATE,SQLERRM;
        RETURN NULL;
    WHEN UNIQUE_VIOLATION THEN
        RAISE WARNING 'audit_trg_function exeception in %.% - UDF ERROR [UNIQUE\]
- SQLSTATE: %, SQLERRM: %',TG_TABLE_SCHEMA,TG_TABLE_NAME,SQLSTATE,SQLERRM;
        RETURN NULL;
    WHEN OTHERS THEN
        RAISE WARNING 'audit_trg_function exeception in %.% - UDF ERROR [OTHER\]
- SQLSTATE: %, SQLERRM: %',TG_TABLE_SCHEMA,TG_TABLE_NAME,SQLSTATE,SQLERRM;
        RETURN NULL;
END;
$BODY$
 LANGUAGE plpgsql VOLATILE
 COST 100;
CREATE TRIGGER AUDIT_TRIGGER BEFORE INSERT OR UPDATE ON booker.ad_decorations
FOR EACH ROW EXECUTE PROCEDURE booker.audit_trg_function();
```

aws professional services

Postgres unexpected behavior

Explicit rollback required on SQLException (when autocommit is off)

- Use savepoints to allow rolling back individual items in the same commit
- Oracle/Mysql have an implicit rollback on SQLException

TIMESTAMP handling

- TIMESTAMP (without time zone) stores the time in the time zone of the client
- Use TIMESTAMP WITH TIME ZONE and don’t provide a time zone to force storing timestamp in UTC

PostgreSQL vs Oracle Play Book

aws professional services

	ORACLE	Postgres
link	Common Data Types	Common Data Types
link	DB hints	Query Planning
link	Inline Views	Inline Views
link	Merge	SQL Merge
link	DBMS Random	Random()
link	DBMS Output	Raise
link	Procedure and Function	Functions
link	OLAP Function	Windowing Function
link	Execute Immediate	Execute
link	Index Organized Table	Postgres Cluster Tables
link	Table Constraints	Table Constraints
link	Table Partitioning including: RANGE, LIST, HASH, COMPOSITE, Automatic LIST	Table Partitioning including: RANGE, LIST
link	Temporary Tables	Temporary Tables
link	Unused Columns	Drop Columns

	Oracle	Postgres
link	Read Only Tables	Read only Roles
link	Index Types	Index Types
link	B Tree Index	B Tree Index
link	Composite Index	Composite Index
link	Bitmapped Index	BRIN Index
link	Functional Index	Expression Index
link	Global and Local Index	Local Index
link	LOB and Secure Files	LOB
link	Materialized Views	Materialized Views
link	Oracle Triggers	Postgres Functional Triggers
link	Views	Views
link	Sequences	Sequences
link	Database Links	Postgres DBlink and FDWrapper

Questions ?

Thank you !