Using TRANLOGOPTIONS EXCLUDETRANS with GoldenGate in Order to Exclude Application Transactions

Edward Whalen
2019-01-10

GoldenGate is designed to replicate very quickly, but tends to work well on smaller transactions. With very large transactions you can see higher lags in the extract, high memory usage by the extract and spikes in lags for the transactions that are waiting on these long transactions. In some cases, it is better to run large purge operations or other cleanup operations on both source and target and avoid replication issues. This can be done by using EXCLUDETRANS and excluding specific named transactions.

Thus, it is desired to be able to control large operations within applications such that they are not replicated with GoldenGate. This capability will allow us to run large jobs on both source and target, thus eliminating the need to replicate, thus reducing replication overhead on very large transactions.

This can be accomplished via the tranlogoptions excludetrans parameter. The tranlogoptions excludetrans parameter works with integrated replicat and allows for the exclusion of named transactions. To accomplish this, add the following to the integrated extract parameter file:

TRANLOGOPTIONS EXCLUDETRANS <transaction name>

For example:

TRANLOGOPTIONS EXCLUDETRANS edstran

Then using the set transaction name ‘edstran’ sql statement, that transaction will be excluded. For example:

Set transaction name ‘edstran’;

<sql>

commit;

This will be excluded. Here is an example that I used to prove that this actually works. I configured an integrated extract, a pump and a replicat (with showsyntax).

I subsequently ran the following SQL statements:

```
SOURCE on GG00A:hr > insert into regions values (62, 'cypress');

1 row created.

SOURCE on GG00A:hr > insert into regions values (63, 'cypress');
```

```
1 row created.

SOURCE on GG00A:hr > commit;

Commit complete.

SOURCE on GG00A:hr > set transaction name 'edstran';

Transaction set.

SOURCE on GG00A:hr > insert into regions values (64, 'cypress');

1 row created.

SOURCE on GG00A:hr > insert into regions values (65, 'cypress');

1 row created.

SOURCE on GG00A:hr > commit;

Commit complete.

SOURCE on GG00A:hr > insert into regions values (66, 'cypress');

1 row created.

SOURCE on GG00A:hr > insert into regions values (67, 'cypress');

1 row created.

SOURCE on GG00A:hr > commit;

Commit complete.
```

The following was observed in the replicat:

```
INSERT INTO "HR"."REGIONS" ("REGION_ID","REGION_NAME") VALUES ('62','cypress')

Statement length: 78

INSERT INTO "HR"."REGIONS" ("REGION_ID","REGION_NAME") VALUES ('63','cypress')

Statement length: 78

INSERT INTO "HR"."REGIONS" ("REGION_ID","REGION_NAME") VALUES ('66','cypress')

Statement length: 78

INSERT INTO "HR"."REGIONS" ("REGION_ID","REGION_NAME") VALUES ('67','cypress')

Statement length: 78
```

Note that the values inserted within the named transaction were not replicated. Thus, by using a named transaction and the integrated extract parameter **TRANLOGOPTIONS EXCLUDETRANS** the transaction is excluded from replication. Thus the overhead of replicating large transactions can be eliminated by running this job on both source and target.