Informix Large Table Operations

by Ben Thompson

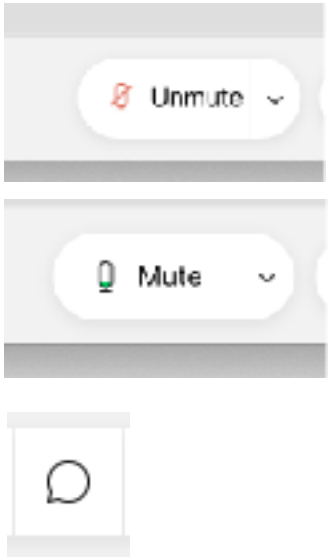
Date: Thursday, May 27, 2021, 2:00pm EDT

Informix Tech Talks by the IIUG

We are launching a new channel on YouTube for Informix Users! This will be a place for Informix how-to videos. More information will be coming soon.

Webcast Guidelines

- **The Webcast is pre-recorded**. The replay and slides will be available on the IIUG Website
- **Please Mute your line.** Background sounds will distract everyone
- **Use the Chat Button** to ask questions



About me

- IT career began in 1999.
- DBA since 2008.
- Database technical lead.
- Writer of the Informed Mix blog.

Characteristics of a “large table operation”?

- Too big to fail (and roll back).
- Acceptable down time window not long enough.
- May be difficult to test.
- May require an imaginative multi-part implementation instead of a more obvious method.

Things that can go wrong

- Long transaction (Long TX) and roll back.
- Taking longer than expected (temptation to hit Ctrl-C).
- Bloated lock table.
- Not enough free space for sort.
- Slow index page logging.
- Slow replication or log backups.

Common large operations

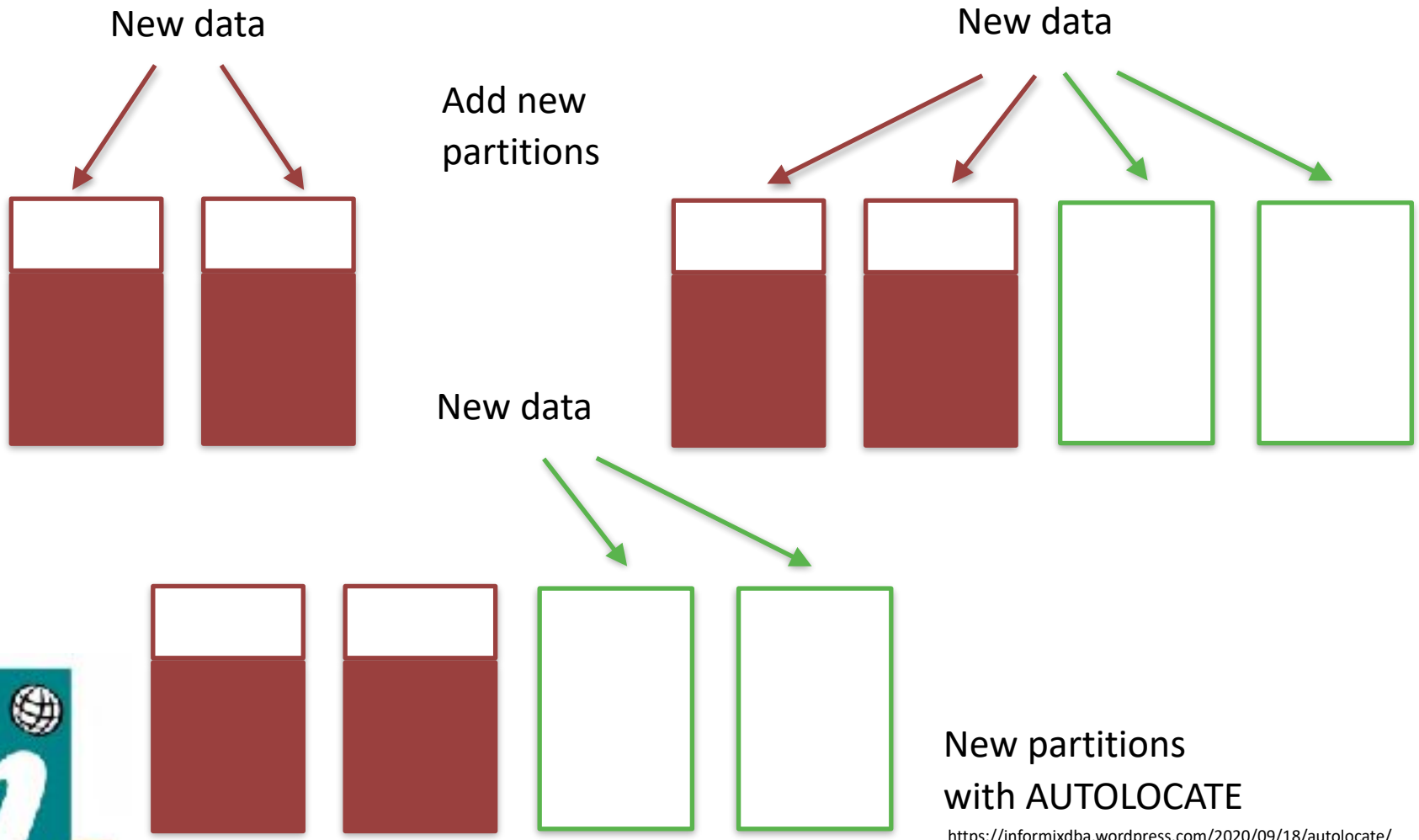
- Building indices
- Table partitioning or re-partitioning
- Copying data

Toolbox

- ALTER FRAGMENT... ADD PARTITION
- CREATE INDEX
- ALTER FRAGMENT ON TABLE… INIT
- ALTER FRAGMENT… ATTACH/DETACH
- Loopback replication
- High performance loader

ALTER FRAGMENT

ALTER FRAGMENT… ADD PARTITION



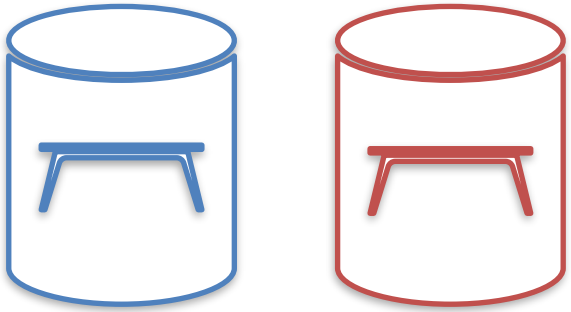
https://informixdba.wordpress.com/2020/09/18/autolocate/

CREATE INDEX

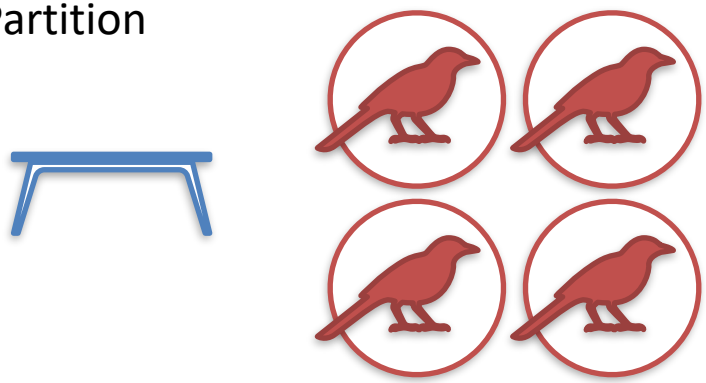
- Previous example affected the table only. Some other functions will implicitly rebuild indices.
- Nothing new here:
  - May wish to set PSORT_NPROCS.
  - Sufficient temporary space (DBSPACETEMP or PSORT_DBTEMP) to avoid partial builds.
  - PDQPRIORITY with enterprise edition.
  - HDR needs to transmit data pages, RSS requires index page logging.

ALTER FRAGMENT ON TABLE… INIT

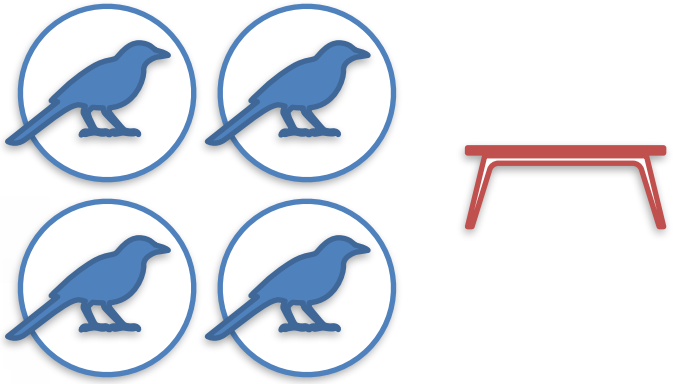
1. Move dbspace



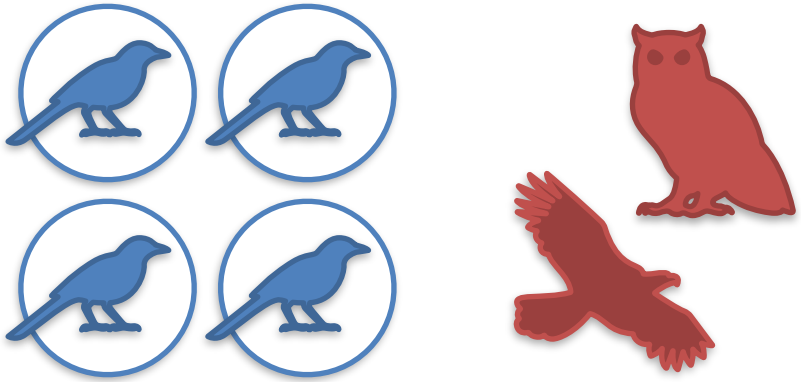
2. Partition



3. De-partition



4. Change partitioning



ALTER FRAGMENT ON TABLE… INIT

The good

- Easy, single statement.
- Preserves all aspects of logical schema.

The bad

- Can be slow on tables with many indices.

The ugly

- Long transaction or Ctrl-C.

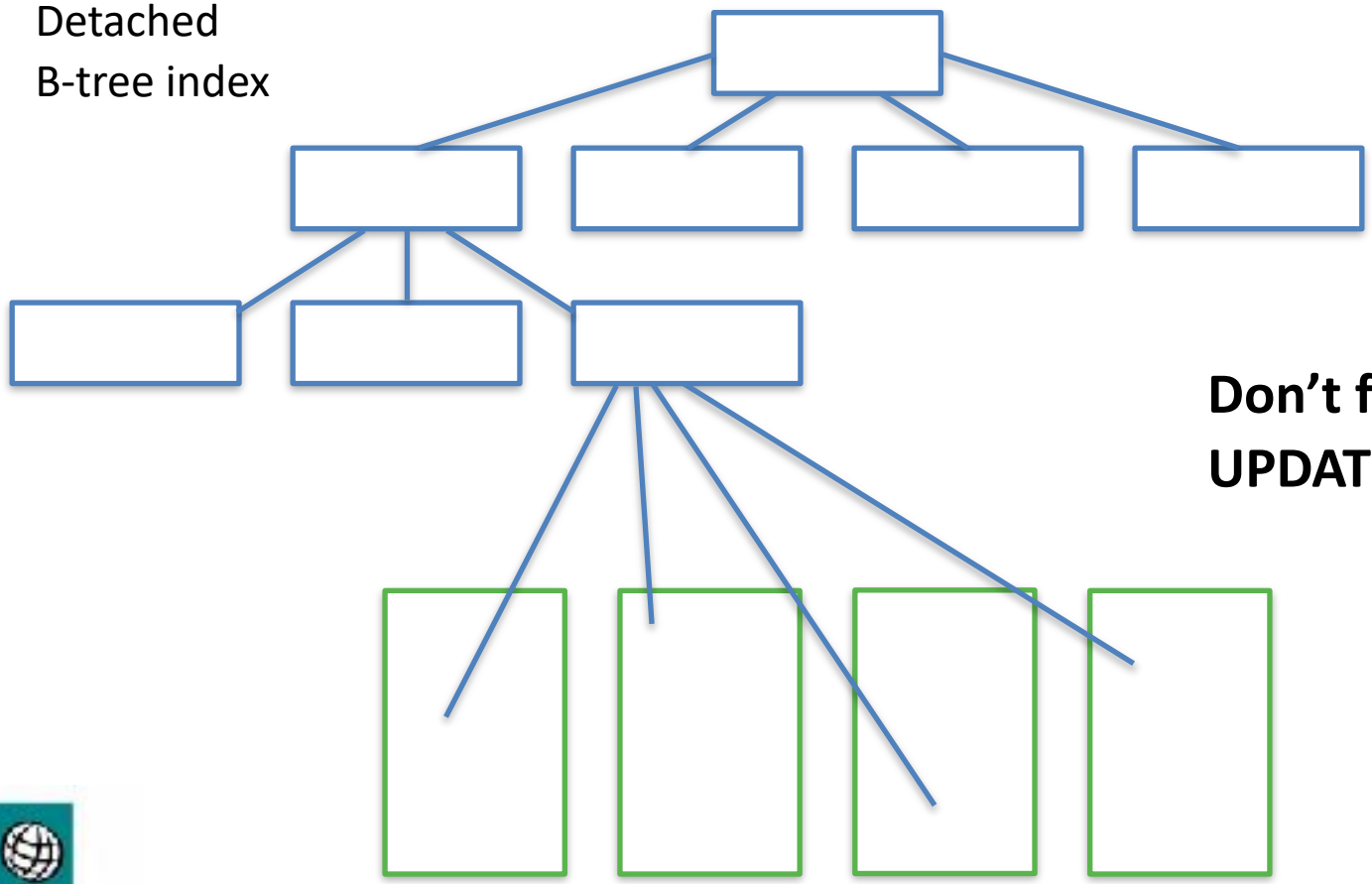
ALTER FRAGMENT ON TABLE… ATTACH

1. Join two non-partitioned tables

2. Join a non-partitioned table to a partitioned table

What about indices?

Detached
B-tree index



Don’t forget to run
UPDATE STATISTICS

Partitioned table

Instant attachment

```
drop table if exists "benthompson".toriginal;
create schema authorization "benthompson"
create table "benthompson".toriginal
  (
    my_id int not null ,
    secondary_id integer default null,
    created datetime year to second default current year to second not null ,
    vchar1 varchar(15) not null ,
    tertiary_id integer not null ,
    expiry_date datetime year to second
  ) in my_dbspace1 lock mode row;

create unique index "benthompson".ioriginal_x1 on "benthompson".toriginal (my_id, vchar1, tertiary_id) using btree in my_dbspace1;
create index "benthompson".ioriginal_x2 on "benthompson".toriginal (secondary_id) using btree in my_dbspace1;
alter table "benthompson".toriginal add constraint primary key (my_id, vchar1, tertiary_id) constraint "benthompson".coriginal_pk  ;

drop table if exists "benthompson".texpression;
create table "benthompson".texpression as select * from toriginal where 1=0;

alter fragment on table texpression init fragment by expression (my_id < 0) in my_dbspace1, (my_id >= 10000000) in my_dbspace2;
OR:
alter fragment on table texpression init fragment by range (my_id) interval (10000000) partition tfm_range_p0 values < 0 in my_dbspace2;

create unique index "benthompson".iexpression_x1 on "benthompson".texpression (my_id, vchar1, tertiary_id) using btree;
create index "benthompson".iexpression_x2 on "benthompson".texpression (secondary_id) using btree;

alter table toriginal drop constraint coriginal_pk;

alter fragment on table texpression attach toriginal as (my_id>=0 and my_id <10000000) after my_dbspace1;
OR:
alter fragment on table texpression attach toriginal as partition texpression_p1 values < 10000000;

alter table "benthompson".texpression add constraint primary key (my_id, vchar1, tertiary_id) constraint "benthompson".cexpression_pk;
```

What about DETACH?

- Similar to ATTACH, all indices must follow the table.

Testing ATTACH/DETACH

- Easy to test these operations with small tables with identical schemas.
- As any operations will be quick with a small table, use ‘oncheck -pe’ to see there is no movement.

```
 openbet_45:'openbet'.texpression                830124       4
 openbet_45:'openbet'.iexpression_x1             830142       4
 openbet_45:'openbet'.iexpression_x2             830146       4
 openbet_45:'openbet'.iexpression_x4             830150       4
 openbet_45:'openbet'.iexpression_x5             830154       4
 openbet_45:'openbet'.iexpression_x6             830158       4
 openbet_45:'openbet'.toriginal                 3808383      12
 openbet_45:'openbet'.toriginal                 4039115      24
 openbet_45:'openbet'.toriginal                 4184015    1152
 openbet_45:'openbet'.toriginal                 4187419    2048
 openbet_45:'openbet'.toriginal                  542155    2048
 openbet_45:'openbet'.toriginal                 1253927    4364
 openbet_45:'openbet'.toriginal                 1482023   40510
 openbet_45:'openbet'.toriginal                 1926691   99800
 openbet_45:'openbet'.toriginal                       3   16384
 openbet_45:'openbet'.ifeedmap2_x1                16387      20
 openbet_45:'openbet'.ifeedmap2_x1                16421   49120
 openbet_45:'openbet'.ifeedmap2_x2                65541   14336
 openbet_45:'openbet'.ifeedmap2_x4               194565   36864
 openbet_45:'openbet'.ifeedmap2_x5               231429   14336
 openbet_45:'openbet'.ifeedmap2_x6               245765   90112

Mon 22 Feb 17:36:34 GMT 2021

Alter fragment completed.

Mon 22 Feb 17:36:34 GMT 2021

 openbet_45:'openbet'.texpression                830124       4
 openbet_45:'openbet'.iexpression_x1             830142       4
 openbet_45:'openbet'.iexpression_x2             830146       4
 openbet_45:'openbet'.iexpression_x4             830150       4
 openbet_45:'openbet'.iexpression_x5             830154       4
 openbet_45:'openbet'.iexpression_x6             830158       4
 openbet_45:'openbet'.texpression               3808383      12
 openbet_45:'openbet'.texpression               4039115      24
 openbet_45:'openbet'.texpression               4184015    1152
 openbet_45:'openbet'.texpression               4187419    2048
 openbet_45:'openbet'.texpression                542155    2048
 openbet_45:'openbet'.texpression               1253927    4364
 openbet_45:'openbet'.texpression               1482023   40510
 openbet_45:'openbet'.texpression               1926691   99800
 openbet_45:'openbet'.texpression                     3   16384
 openbet_45:'openbet'.iexpression_x1              16387      20
 openbet_45:'openbet'.iexpression_x1              16421   49120
 openbet_45:'openbet'.iexpression_x2              65541   14336
 openbet_45:'openbet'.iexpression_x4             194565   36864
 openbet_45:'openbet'.iexpression_x5             231429   14336
 openbet_45:'openbet'.iexpression_x6             245765   90112
```

LOOPBACK REPLICATION

Loopback replication - method

- Loopback replication will replicate new transactions; it will not automatically sync tables.
- Create an empty copy of the table.
- Bulk copy the data.
- Create indices and primary keys.
- Define and start replicates.
- Check replicates for inconsistencies and repair.

Loopback replication - preparation

- Define CDR_DBSPACE, CDR_QHDR_DBSPACE and CDR_QDATA_SBSPACE and create corresponding dbspaces and an sbspace.
- Create an ER group and pseudo loopback group in sqlhosts and onconfig variable DBSERVERALIASES

  https://www.ibm.com/docs/en/informix-servers/12.10?topic=replication-loopback-configuration
- Define ‘ER server’ and ‘loopback server’ using 'cdr define server’.

Loopback replication - tuning

Parameter	Value	Comment
CDR_EVALTHREADS	At least one per CPU VP but < 10 e.g. 0,10	Manual states at least one per CPU VP but I advise against this for large systems
CDR_QUEUEMEM	64 - 128 MB i.e. 65536 - 131072	Value recommended by HCL support.
CDR_SUPPRESS_ATSRISWARN	for loopback replication: 3	Converts updates where no row is present to inserts
CDR_QDATA_SBSPACE	One or more smart blob spaces	May need these to be quite large for big loopback jobs, up to 100 GB

Recommendations are for versions prior to ER improvements coming in 14.10.xC6 or 14.10.xC7. Use at your own risk and do your own testing!

Bulk data load

- More than one way to do this. I like high performance loader (HPL) because:
  - can transmit data over network as a stream without intermediate files.
  - handles data commits.
  - no need to write any code.

```
mknod /home/informix/toriginal.pipe p
onpladm create job toriginal_unload -d 'cat > /home/informix/toriginal.pipe' -fup -D my_dbname
-t toriginal -T my_instance_tcp
onpladm create job texpression_load -d 'cat /home/informix/toriginal.pipe' -flpc -D my_dbname
-t texpression -T my_instance_tcp
onpladm run job toriginal_unload -fu -l /home/informix/toriginal_unload.out -S my_instance_tcp
onpload -j texpression_load -fl -I 200000 -i 200000 -l /home/informix/texpression_load.out
```

Loopback replication - define

```
cdr define server -A /var/informix/ats/
g_my_instance_er_server -R /var/informix/ris/
g_my_instance_er_server -I g_my_instance_er_server

cdr define server -A /var/informix/ats/
g_my_instance_loopback -R /var/informix/ris/
g_my_instance_loopback -I g_my_instance_loopback -S
g_my_instance_er_server

cdr define repl --connect=g_my_instance_er_server
loopback_replicate --conflict=always --scope=row --
ats --ris --floatieee --
master=g_my_instance_er_server "P
my_dbname@g_my_instance_er_server:benthompson.origin
al_table" "select * from original_table" "R
my_dbname@g_my_instance_loopback:benthompson.new_tab
le" "select * from new_table"
```

Loopback replication - repair

Check and repair replicate:

```
nohup cdr check replicate –
master=g_my_instance_er_server --repl=
loopback_replicate g_my_instance_loopback --repair
>loopback_replicate.out --name
loopback_replicate_check_repair 2>&1 &
```

Check status with:

```
cdr stats check
```

Monitor ER with:

```
cdr view profile
```

Loopback replication - final steps

- Stop and delete replicate.
- Switch tables, may include:
  - Drop foreign keys on original table.
  - Rename constraints and indices.
  - Rename table.
  - Drop and recreate triggers.
  - Recreate any referencing foreign keys, find with:

```
select t.tabname, c.constrname from sysconstraints c,
systables t where c.tabid=t.tabid and c.constrid in
(select constrid from sysreferences where ptabid in
(select tabid from systables where tabname='XXXXXXXX'));
```

Thank You

Informix Tech Talks by the IIUG on YouTube

We have launched a new channel on YouTube for Informix Users! Please subscribe to our channel on YouTube to stay informed. This will be a place for Informix how-to videos.

Subscribe at: https://www.youtube.com/c/InformixTechTalksbytheIIUG