ASSURING BUSINESS AND SERVICES.

BE ASSURED.

Tuesday 25th – Thursday 27th October 2005

NEW YORK CITY 2005 Netcool® Users Conference

Netcool/OMNIbus V7 Automation Clinic

James E Brunke
Chief Technologist
Crystal Technology Solutions Group Inc
jbrunke@ctsgi.com
http://www.ctsgi.com/

Welcome

- Brief overview of Procedural SQL specific to automations
- Explanation of different Automation Trigger types
- Calculating Duration
- Deduplication Clearing
- “X in Y” Event Escalation
- Lots of code to take home and try (YMMV)

Procedural SQL

- INSERT, UPDATE, DELETE
- WRITE INFO (CREATE FILE)
- ALTER (TRIGGER, SYSTEM)
- RAISE SIGNAL
- EXECUTE, CALL
- Programming constructs (SET, IF THEN ELSE, CASE WHEN, FOR EACH ROW, FOR, BREAK, CANCEL)
- Functions – old ones (getdate), lots of new ones (substr, to_time…)

Procedural SQL (cont)

- User Variables (%user.user_id, %user.user_name, %user.app_name…)
- Trigger Variables (%trigger.xxxxx)
- Operators (added %= and %!= to ignore case, %<, %<=, %>, %>= alphabetic order)
- No SELECT inside trigger (Evaluate clause)

Trigger Types

- Temporal
- Signal
- Database (woo-hoo!!!)

Temporal Triggers

- Operate fundamentally the same as in Omnibus 3.x
- Priority indicates what order to run (previously in alpha order)
- Indicate how often to run (every xx seconds, minutes, etc)
- Evaluate contains SELECT statement if procedure is to operate using data from a specific set of rows (FOR EACH ROW)
- You can’t specify ORDER BY in SELECT

Signal Triggers

- Trigger fires when signal received
- System triggers include things like startup, shutdown, client (dis)connection, logins, backups, etc)
- Can also create user triggers which can be kicked off with RAISE SIGNAL statement
- Variables are defined (%signal.xxxx) which contain attributes specific to the signal trigger raise

Database Triggers

- Without question, the most powerful addition made in V7 (IMHO)
- Trigger will fire when either before or after a modification (insert, update, delete, reinsert) is attempted to an ObjectServer table
- Trigger can fire for every row modified or just once for the specific SQL statement executed
- If WHEN clause is specified, trigger will only fire when condition is true
- CANCEL statement in the procedure section stops operation from occurring in a before database trigger
- No Evaluate clause
- old. and new. Variables

Database Triggers (cont)

Operation	*Timing*	*new.<field> Available?*	*new.<field> Modifiable?*	*old.<field> Available?*	*old.<field> Modifiable?*
INSERT	BEFORE	Yes	**Yes**	No	No
INSERT	AFTER	Yes	No	No	No
UPDATE	BEFORE	Yes	**Yes**	Yes	No
UPDATE	AFTER	Yes	No	No	No
DELETE	BEFORE	No	No	Yes	No
DELETE	AFTER	No	No	Yes	No
REINSERT	BEFORE	Yes	No	Yes	**Yes**
REINSERT	AFTER	Yes	No	No	No

Duration Calculation

- Concept presented at the 2004 NUC in session “Using Netcool To Create Availability Reports”
- Use Generic Clear to calculate amount of time between Problem and Resolution events
- LastCleared field in Problem event will contain date/time stamp indicating when it was cleared
- Duration field will contain “rolled up” time in seconds that the Problem lasted
- Handles deduplication

Duration Calculation prerequistes

alerts.problem_events2

Name	Data Type	Length	Primary Key
Identifier	VarChar	255	X
LastOccurrence	UTC	4	
AlertKey	VarChar	255	
AlertGroup	VarChar	64	
Node	VarChar	64	
Manager	VarChar	64	
Resolved	Boolean	4	
ClearedTime	***UTC***	***4***	
Duration	***Integer***	***4***	

Trigger generic_clear

```
-- Populate a table with Type 1 events corresponding to any
   uncleared Type 2 events

for each row problem in alerts.status where problem.Type = 1
   and problem.Severity > 0 and (problem.Node +
   problem.AlertKey + problem.AlertGroup + problem.Manager)
   in ( select Node + AlertKey + AlertGroup + Manager from
   alerts.status where Severity > 0 and Type = 2 )

begin

    insert into alerts.problem_events2 values
      (problem.Identifier, problem.LastOccurrence,
      problem.AlertKey, problem.AlertGroup, problem.Node,
      problem.Manager, false, problem.ClearedTime,
      problem.Duration );

end;
```

Trigger generic_clear (cont 1)

```
-- For each resolution event, mark the corresponding
   problem_events entry as resolved and clear the resolution

for each row resolution in alerts.status where
   resolution.Type = 2 and resolution.Severity > 0

begin

   set resolution.Severity = 0;

   update alerts.problem_events2 set Resolved = true,
   ClearedTime = resolution.LastOccurrence, Duration =
   Duration + (resolution.LastOccurrence – LastOccurrence)
   where LastOccurrence < resolution.LastOccurrence and
   Manager = resolution.Manager and Node = resolution.Node
   and AlertKey = resolution.AlertKey and AlertGroup =
   resolution.AlertGroup ;

end;
```

Trigger generic_clear (cont 2)

```
-- Clear the resolved events

for each row problem in alerts.problem_events2 where
  problem.Resolved = true

  begin

     update alerts.status via problem.Identifier set
     Severity = 0, ClearedTime = problem.ClearedTime,
     Duration = problem.Duration;

  end;

-- Remove all entries from the problems table

delete from alerts.problem_events2;
```

Deduplication Clearing

- Deduplication clearing useful when your ObjectServer is very busy
- Could be done in Omnibus V3.x but Tally would increase by one for each Problem and Resolution
- Can be done just for events that are “heavy hitters”
- There will be no separate Resolution event to track
- Pre-requisites
  - Identifiers of Problem and Resolution events must match
  - Severity of Resolution event should be zero
- Example also includes code to populate LastCleared and Duration fields

Trigger deduplication

- **Database trigger, before reinsert on alerts.status**

```
set old.StateChange = getdate();
set old.InternalLast = getdate();
if ((old.Type = 1) and (new.Type = 2)) then
    set old.LastCleared = new.LastOccurrence;
    set old.Duration = old.Duration + (new.LastOccurrence –
      old.LastOccurrence);
    set old.Severity = 0;
```

Trigger deduplication (cont)

```
else
   set old.Tally = old.Tally + 1;
   set old.LastOccurrence = new.LastOccurrence;
   set old.Summary = new.Summary;
   set old.AlertKey = new.AlertKey;
   if (( old.Severity = 0) and (new.Severity > 0)) then
      set old.Severity = new.Severity;
   end if;
end if;
```

X in Y escalation

- "I want something to happen (change severity, notification, electric shock to the chair, etc) when an alarm happens X times in Y seconds"
- Sliding window - watching Tally, FirstOccurrence, LastOccurrence - not good enough!!!
- Previously only could be done with "other" product (Impact, etc)
- Why is this so hard? Deduplication!!
- Fundamentally, we need to UN-Deduplicate events so we can see if a particular event has exceeded it's X in Y threshold

X in Y escalation prerequisites

- Add new fields in alerts.status table
  - xEvents [Integer] – How many events needed to trigger escalation
  - ySeconds [Integer] – Duration of time necessary to trigger escalation
- Probe/Monitor rules populate xEvents and ySeconds fields for specific events
- Create new table called alerts.xiny with following fields
  - Identifier [VarChar (255)] – Primary Key – Corresponds to Identifier of event in alerts.status
  - ExpireUTC [UTC] – Primary Key – When will this row will no longer be needed in the table?
- Four database triggers and one temporal trigger

Trigger xiny_insert

- **Database trigger, before insert on alerts.status**
- **When (new.xEvents > 0) and (new.ySeconds > 0) and (new.Severity > 0)**

```
declare
    xinycount int;
begin
   insert into alerts.xiny values (new.Identifier,
     (new.LastOccurrence + new.ySeconds));
   set xinycount = 0;
```

Trigger xiny_insert (cont)

```
   for each row tmprow in alerts.xiny where
     tmprow.Identifier = new.Identifier and tmprow.ExpireUTC
     >= getdate()

   begin

      set xinycount = xinycount + 1;

   end;

   if (xinycount >= new.xEvents) then

      set new.SuppressEscl = 2;

   end if;

end
```

Trigger xiny_reinsert

- **Database trigger, before reinsert on alerts.status**
- **When (new.xEvents > 0) and (new.ySeconds > 0) and (new.Severity > 0) and (SuppressEscl <> 2)**

```
declare

    xinycount int;

begin

   insert into alerts.xiny values (old.Identifier,
     (new.LastOccurrence + new.ySeconds));

   set xinycount = 0;
```

Trigger xiny_reinsert (cont)

```
   for each row tmprow in alerts.xiny where
     tmprow.Identifier = old.Identifier and tmprow.ExpireUTC
     >= getdate()

   begin

      set xinycount = xinycount + 1;

   end;

   if (xinycount >= new.xEvents) then

      set old.SuppressEscl = 2;

   end if;

end
```

Trigger xiny_update

- **Database trigger, before update on alerts.status**
- **When (new.xEvents > 0) and (new.ySeconds > 0) and (SuppressEscl <> 2)**

```
declare
    xinycount int;
begin
   set xinycount = 0;
   for each row tmprow in alerts.xiny where tmprow.Identifier =
     new.Identifier and tmprow.ExpireUTC >= getdate()
   begin
      set xinycount = xinycount + 1;
   end;
```

Trigger xiny_update (cont)

```
   if (xinycount >= new.xEvents) then
      set new.SuppressEscl = 2;
   end if;
end
```

Trigger xiny_delete

- **Database trigger, before delete on alerts.status**
- **When (old.xEvents > 0) and (old.ySeconds > 0)**

```
begin

   delete from alerts.xiny where Identifier =
     old.Identifier;

end
```

Trigger xiny_expire

- **Temporal trigger, Every 30 seconds**

```
begin
   for each row tmprow in alerts.xiny where tmprow.ExpireUTC
     < getdate()
   begin
      update alerts.status via tmprow.Identifier set
        StateChange = getdate();
      delete from alerts.xiny where Identifier =
        tmprow.Identifier;
   end;
end
```

X in Y Example

Escalate “Threshold” event when I receive four events in 2 minutes

Ruleset will assign xEvents = 4, ySeconds = 120

- **Event received at 00:00:00. Identifier = “Threshold”, Tally = 1**
- **Trigger xiny_insert creates new row in alerts.xiny. Identifier = “Threshold”, ExpireUTC = 00:02:00 (row count = 1)**
- **Event received at 00:00:25. Identifier = “Threshold”, Tally = 2**
- **Trigger xiny_reinsert creates new row in alerts.xiny. Identifier = “Threshold”, ExpireUTC = 00:02:25 (row count = 2)**
- **Event received at 00:01:58. Identifier = “Threshold”, Tally = 3**
- **Trigger xiny_reinsert creates new row in alerts.xiny. Identifier = “Threshold”, ExpireUTC = 00:03:58 (row count = 3)**

X in Y Example (cont)

- Trigger xiny_expire fires at 00:02:00 and deletes row Identifier = “Threshold”, ExpireUTC = 00:02:00 in alerts.xiny (row count = 2)
- Event received at 00:02:08. Identifier = “Threshold”, Tally = 4
- Trigger xiny_reinsert creates new row in alerts.xiny. Identifier = “Threshold”, ExpireUTC = 00:04:08 (row count = 3)
- Event received at 00:02:14. Identifier = “Threshold”, Tally = 5
- Trigger xiny_reinsert creates new row in alerts.xiny. Identifier = “Threshold”, ExpireUTC = 00:04:08 (row count = 4). Trigger also sets SuppressEscl = 2 since X in Y threshold has been reached

X in Y Improvements

- Won't work for sub-second events
- De-escalation
- Multiple escalation tiers
- Write journal events to event when escalated – would require addition of serial to alert.xiny table since that is key to alerts.journal table
- Measure performance impact on Object Server
- Use of aggregate selects if/when supported in database triggers

Conclusions

- Extremely powerful features available in V7 Automations
- Requires “Thinking out of the box”

Crystal Technology Solutions Group Inc can help with your Omnibus V7 installation/upgrade

Contact us!!

Questions?