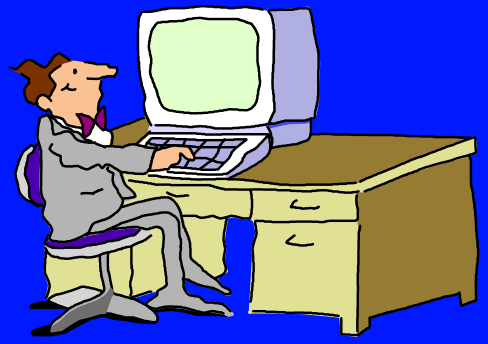
# Fine Calculations via PL/pgSQL

## A Fine Generator Replacement

# Alpha-G Consulting

- Data Migration
- System Administration
- Support
- Hosting
- Reporting/Data Transfer & Interchange
- Contact john@alphagconsulting.com
- Slides: http://alphagconsulting.com/EvgConf2011/



# The Project: Fine Generation

- Library charges $1 per day
- But:
  - Max fine is price of item; e.g.: $14.25
  - Not getting reliable skipping of closed days
  - Need an easy way to fix fines that are wrong
- Easier to DIY in PL/pgSQL than other options

# Using PL/pgSQL

- Need a SQL script? Write a function.
- Documentation not much help:
  - Lacks tutorial approach to advanced topics
  - Still based on old requirements of much earlier versions of Postgres
- Useful patterns emerged from
  - Writing
  - Modifying
  - Testing

# Building the Fines Function

- Critical Calculation: Days between Dates
- What about Closed Days?
  - No built-in function available
  - Creating a new SQL function
- Overall Structure of Overdue Calculation
- What it would need to be more general
- Patterns and Suggestions

# Days Between Dates

```
select ac.id
     , date_trunc( 'day'
                 , now() - ac.due_date
                 )
  from action.circulation ac
 where ac.due_date < now()
;
```

# What about Closed Days?

- No built-in function in Postgres
- Create one
- Closed Days
  - actor.hours_of_operation entries
  - Use entries for action.circulation.circ_lib
- Determine the day of week for:
  - Run date (today)
  - Due date
- Check day of week for each day in range

# Which Days Closed?

```
create view alpha_g.closed_day_of_week
    as
select id
     , 0 as "dow"
     , (   dow_0_open   = '00:00:00'
       and dow_0_closed = '00:00:00'
       ) as "is_closed"
  from actor.hours_of_operation
 union
select id
     , 1 as "dow"
     , (   dow_1_open   = '00:00:00'
       and dow_1_closed = '00:00:00'
       ) as "is_closed"
  from actor.hours_of_operation
...
```

# Data from closed_day_of_week

| shortname | dow | is_closed |
|---|---|---|
| MAIN | 0 | false |
| MAIN | 1 | false |
| MAIN | 2 | false |
| MAIN | 3 | false |
| MAIN | 4 | false |
| MAIN | 5 | false |
| *MAIN* | *6* | *true* |
| CMB | 0 | false |
| CMB | 1 | false |
| CMB | 2 | false |
| CMB | 3 | false |
| CMB | 4 | false |
| CMB | 5 | false |
| *CMB* | *6* | *true* |

# is_closed_day Function

```
create function alpha_g.is_closed_day
                ( org_unit_id_param int
                , the_date_param    timestamptz
                ) returns boolean
    as $$
declare  closed_var boolean := false ;
begin
  select is_closed into closed_var
    from alpha_g.closed_day_of_week
   where id  = org_unit_id_param
     and dow = extract
                 ( ISODOW from the_date_param )
                 - 1 ; -- gives Mon 0 - Sun 6
  return closed_var ;
end ;
$$ language plpgsql ;
```

# Adding Logic: org_unit_closed

```
-- ... check actor.hours_of_operation
if ( closed_var ) then
  return closed_var ;
end if ;
-- check actor.org_unit_closed
select exists
       ( select 1
           from actor.org_unit_closed
          where org_unit = org_unit_id_param
            and   ( the_date_param
                  , interval '0 days' )
                  overlaps
                  ( close_start, close_end )
       )
  into closed_var ;
 return closed_var ;
```

# open_days_between Function

```
create function alpha_g.open_days_between
                ( first_date_param  timestamptz
                ,  last_date_param  timestamptz
                ) returns int
    as $$
declare  one_day_var      timestamptz       ;
         end_day_var      timestamptz       ;
         open_days_var    int          := 0 ;
begin
  /* check each day in range and count up
     days library is not closed
  */
  return open_days_var ;
end ;
$$ language plpgsql ;
```

# open_days_between BEGIN/END

```
begin
  one_day_var
    = ( date_trunc( 'day', first_date_param )
      + '1 day' ); -- day after 1st day
  end_day_var
    =   date_trunc( 'day',  last_date_param );
  while ( one_day_var <= end_day_var ) loop
    if ( not alpha_g.is_closed_day
                      ( org_unit_id_param
                      , one_day_var ) ) then
      open_days_var := open_days_var + 1 ;
    end if ;
    one_day_var = one_day_var + '1 day' ;
  end loop ;
  return open_days_var ;
end ;
```

# Structure: generate_fines

- Parameters
  - Run Date
  - Grace Period
- Main loop
  - Select entries from action.circulation
    where due_date < ( Run Date - Grace Period )
  - Recheck counting only open days
    - If fine owed
      Insert billing

# Declarations: generate_fines

```
create function alpha_g.generate_fines
                ( run_date_param  date
                , grace_per_param interval
                ) returns integer
    as $$
declare
    count_var          integer := 0;
    grace_days_var     integer ;
    today_var          date    ;

    circ_id_var        bigint  ;
    org_unit_var       integer;
    due_date_var       date    ;
    fine_rate_var      numeric( 6, 2 );
    max_fine_var       numeric( 6, 2 );
    billed_amt_var     numeric( 6, 2 );
```

# Declarations *(continued)* & Initializations

```
    ...
    days_overdue_var integer;
    new_billing_var  numeric( 6, 2 );
    reached_max_var  boolean;
begin
    -- set the value to be used for run
    if ( run_date_param = null ) then
        today_var := current_date;
    else
        today_var := run_date_param;
    end if ;
    -- calculate the grace period days
    grace_days_var
      := cast( extract( 'days'
                        from grace_per_param )
            as integer );
```

# Main Loop: SELECT

```
for         circ_id_var, org_unit_var, due_date_var
          , fine_rate_var, max_fine_var, billed_amt_var
 in select
            ac.id        , ac.circ_lib , ac.due_date
          , ac.recuring_fine, ac.max_fine
          , coalesce( ( select sum( amount )
                          from money.billing
                         where xact  = ac.id
                           and btype = 1 -- overdues
                      )
                    , 0.00
                    )
       from action.circulation ac
      where ac.checkin_time is null
        and ac.stop_fines   is null
        and date_trunc( 'day', ac.due_date )
          < ( today_var - grace_per_param )
        and ac.xact_finish  is null -- just in case
loop
```

# Main Loop: Check Open Days

```
for
    ...
loop
  -- not w/in grace period by simple condition
  -- in WHERE clause, check more carefully
  days_overdue_var
    := alpha_g.open_days_between( org_unit_var
                                , due_date_var
                                , today_var
                                );

  -- within check grace period
  -- (accounting for closed days)?
  if ( days_overdue_var <= grace_days_var ) then
      continue ;
  end if ;

  ...
```

# Main Loop: Fine Amount

```
...
loop
    ...
  -- past grace period:fines owed
  -- (take into account what already billed)
  new_billing_var
    := ( days_overdue_var * fine_rate_var )
       - billed_amt_var ;
  -- defensive programming--no negative billings
  if ( new_billing_var <= 0.00 ) then
     continue ; -- skip it
  end if ;
  -- check to ensure not over max fine
  if ( new_billing_var >= max_fine_var ) then
    new_billing_var := max_fine_var - billed_amt_var ;
    reached_max_var := true ;
  else
    reached_max_var := false ;
  end if ;
  ...
```

# Main Loop: Insert Billing

```
...
 -- insert billing
 insert into money.billing
       ( xact, amount, billing_type, btype, note )
 -- allowing billing_ts to default to now()
 select circ_id_var, new_billing_var
       , cbt.name, cbt.id
       , 'Auto-generated Overdue fine (ag script)'
      || run_date_param
   from config.billing_type cbt
  where cbt.id = 1 ;

 if ( reached_max_var ) then
   update action.circulation
      set stop_fines      = 'MAXFINES'
        , stop_fines_time = now()
    where id = circ_id_var ;
 end if ;
...
```

# Main Loop: End & Function Return

```
  ...
    -- insert into tracking table (debugging aid)
    insert into alpha_g.generated_fine_test
          ( xact          , run_date  , days_overdue
          , fine_amount     , is_max_fine      )
    values
          ( circ_id_var, today_var, days_overdue_var
          , new_billing_var, reached_max_var ) ;

    -- keep track of how many fines generated
    count_var := count_var + 1 ;

  end loop ;

  return count_var ;
end ;
$$ LANGUAGE plpgsql ;
```

# Generalizing the Function

- Handle Different Fine Intervals
  - Currently assumes per day fine interval

```
-- calculate the grace period days
grace_days_var
   := cast( extract( 'days'
                     from grace_per_param )
         as integer );
```

  - Unfortunately, not a supported operation:

```
( current_time - due_date_var )
/ fine_interval_var
```

# Normalize all Intervals

- Cannot retrieve unit from interval nor base value
- Switch all intervals to minutes?

```
create or replace function alphag.interval_as_minutes
                           ( timeInterval interval )
                  returns  bigint
    as
$$
declare jInterval interval ;
        asMinutes bigint   ;
begin
    jInterval := justify_interval( timeInterval ) ;
    asMinutes := ( extract( months  from interval ) * 43200 )
               + ( extract( days    from interval ) *  1440 )
               + ( extract( hours   from interval ) *    60 )
               + ( extract( minutes from interval ) ;
    return asMinutes ;
end ;
$$ LANGUAGE plpgsql ;
```

# Patterns and Suggestions

- Helpful to Distinguish
  - Parameters

    grace_per_param  p_grace_per
  - Variables

    grace_days_var  v_grace_days
- Explain yourself to yourself (and others)

```
-- check actor.org_unit_closed
```

- Consider implementing test version
  - Often effective to add restriction on main SELECT
  - Added action.circulation.id parameter

# generate_fines_test

```
create function alpha_g.generate_fines
                ( run_date_param  date
                , grace_per_param interval
                , xact_id_param   bigint
                ) returns integer

...
for         circ_id_var, ...

 in select ac.id         , ...
      from action.circulation ac
     where ac.checkin_time is null
       and ac.stop_fines   is null
       and date_trunc( 'day', ac.due_date )
         < ( today_var - grace_per_param )
       and ac.xact_finish  is null -- just in case
       and ac.id = xact_id_param
```

SELECT now returns a single row. All other logic is identical.

# Capture Values in RECORD

```
create function alpha_g.generate_fines
                ( run_date_param  date
                , grace_per_param interval
                ) returns integer
    as $$
declare
    od_row              record ;
    ...
for         od_row  -- holds values for all columns
 in select
            ac.id       , ac.circ_lib , ac.due_date
          , ac.recuring_fine, ac.max_fine
          , coalesce( ( select sum( amount )
                          from money.billing
                         where xact  = ac.id
                           and btype = 1 ) -- overdues
                    , 0.00
                    ) as "billed_amount"
      from action.circulation ac
```

# Using RECORD Fields

```
-- using individual variables

  days_overdue_var
    := alpha_g.open_days_between( org_unit_var
                                , due_date_var
                                , today_var
                                );

-- using row variable's fields

  days_overdue_var
    := alpha_g.open_days_between( od_row.circ_lib
                                , od_row.due_date
                                , today_var
                                );
```

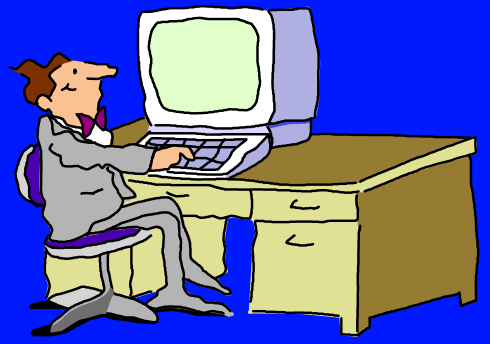
Record variable dot column name replaces many declared variables

# Interactive Debugging?

- Supposed to be possible
- Setup on server side is complex
  - Many questions about it in Postgres forums
  - Unable to find clear, step-by-step instructions
  - If anyone figures it out, and it works well, I'd like to hear about it

# Questions and Discussion

# Alpha-G Consulting



- Data Migration
- System Administration
- Support
- Hosting
- Reporting/Data Transfer & Interchange
- Contact john@alphagconsulting.com
- Slides: http://alphagconsulting.com/EvgConf2011/