Writing an “Enterprise Scheduler” using Perl, MySQL, SSH, and POE

Albert P. Tobey
Pittsburgh Perl Workshop
September 23, 2006

http://www.tobert.org
tobert@gmail.com

Thank you to:

Developers, Advisors, and Testers:

Aaron Nienhuis
Keith Sederholm
Rick Siner
Joel Meulenberg
Keith Brunsting
Jeff Klein

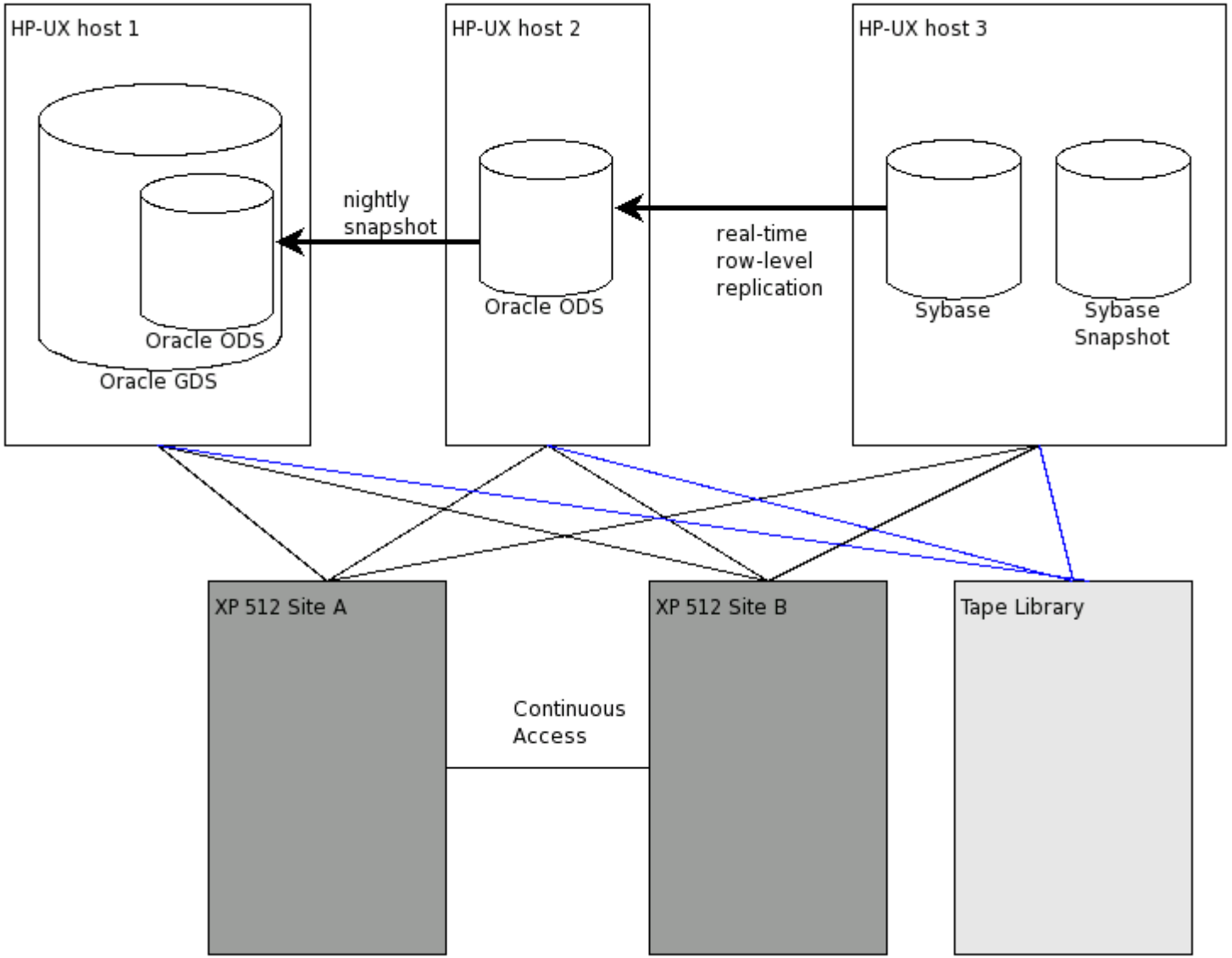
IS Leadership:

Jim Slubowski
Mike Trus
Krischa Winright
Rick Siner

Priority Health is often looking for talented Perl and PL/SQL developers.
http://www.priorityhealth.com/corporate/careers/opportunities/

The Problem

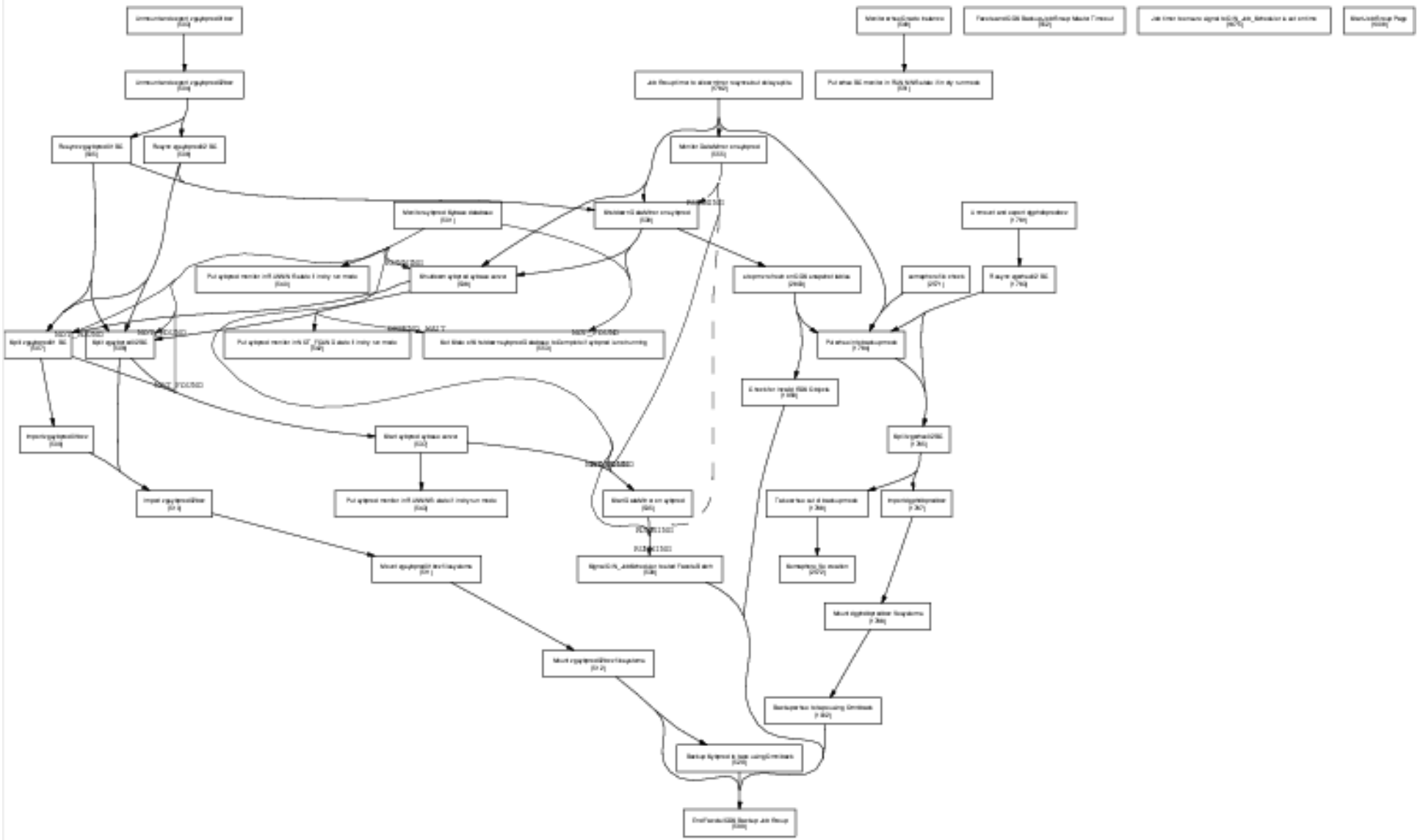
- We upgraded our ERP-like system in 2002.
- It runs Sybase.
- That's fine, but we're an Oracle shop.
- Replication!
- Sybase -> Datamirror -> Oracle
- Now back it up, smart guy.



ODS: Operational Data Store
GDS: Global Data Store

Process Overview

- kick all users out of FACETS
- quiesce all transactions in Sybase
- stop replication
- shut down Sybase
- put the Oracle ODS in hot backup mode
- start BCV copy operation on the Sybase volumes (disk snapshot)
- export the ODS snapshot from the GDS
- export the ODS volume group from the GDS host
- start BCV copy operation to update the ODS snapshot
- put the GDS into hot backup mode
- start BCV copy operation on the GDS
- start backup of each BCV after copy is complete
- after ODS backup completes:
  - import the volume group on the GDS host
  - rename the oracle instance
  - start Oracle
  - export relevant tables
  - import tables into the GDS read-only
- start Sybase
- start the ODS
- start replication
- once replication is complete, make Sybase available to users



Solution Requirements

- sane dependency management
- parallelize as widely as possible
- run arbitrary code on multiple hosts
- centralized management
- configuration separate from program
- extensive logging
- ability to notify administrators of status and incidents

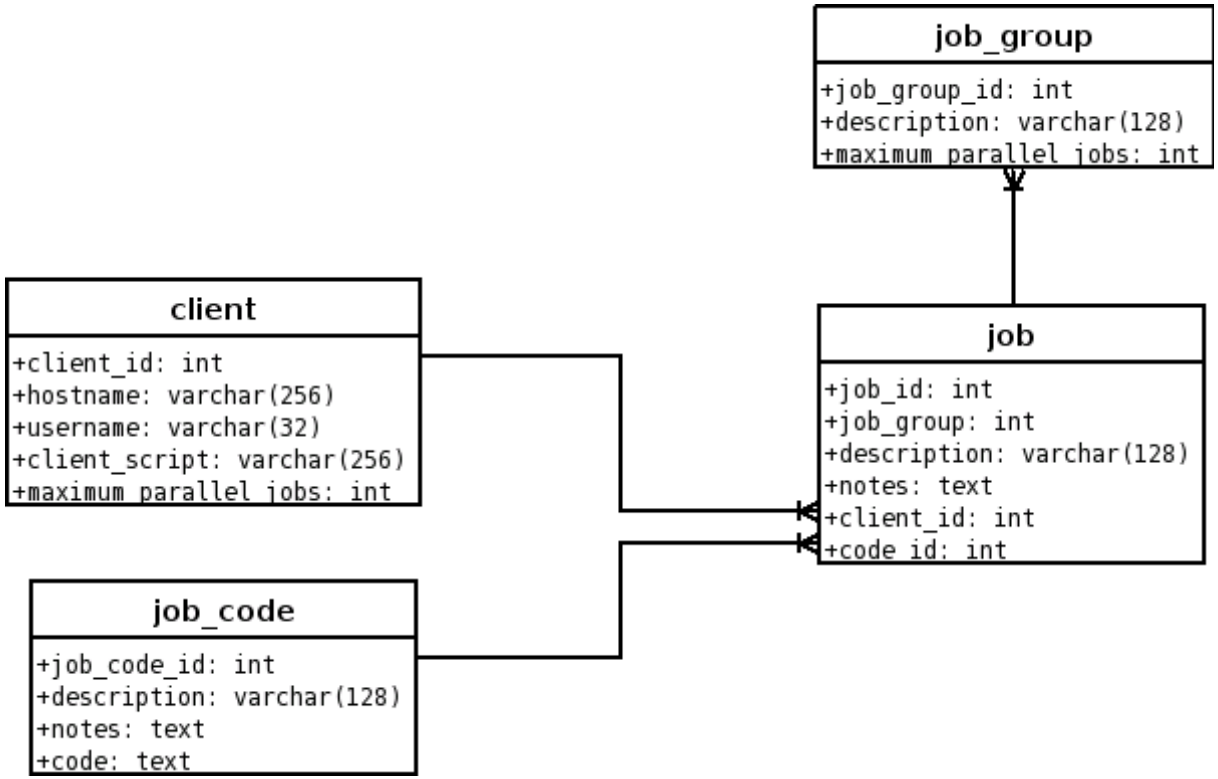
Tools

- Perl
  - rapid prototype/development
  - also great for production
- SSH
  - well-understood and available
- MySQL
  - I knew it better than Oracle at the time
- POE (Perl Object Environment)
  - Handy for multiplexing lots of things in a single thread

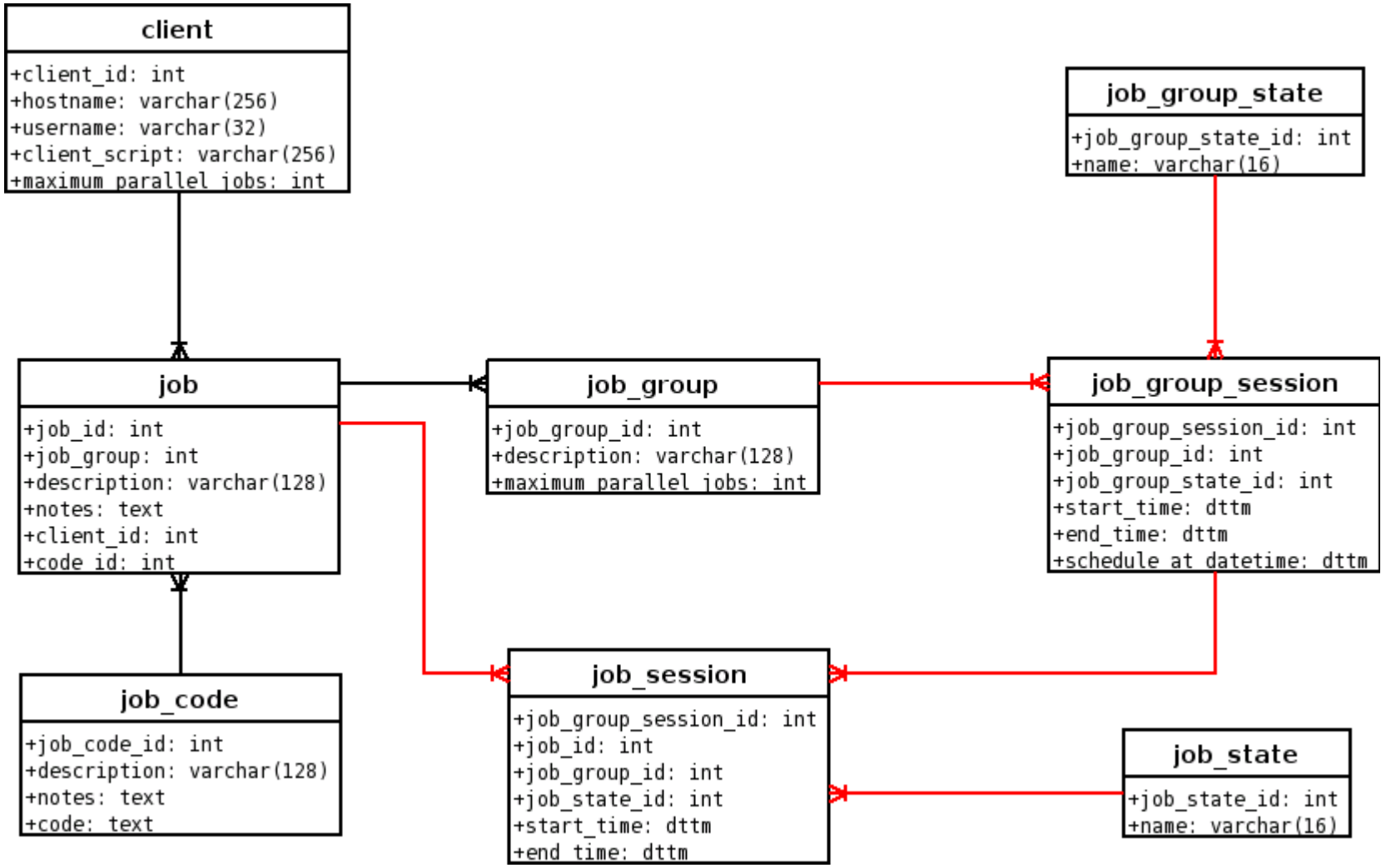
Dependency Graphs

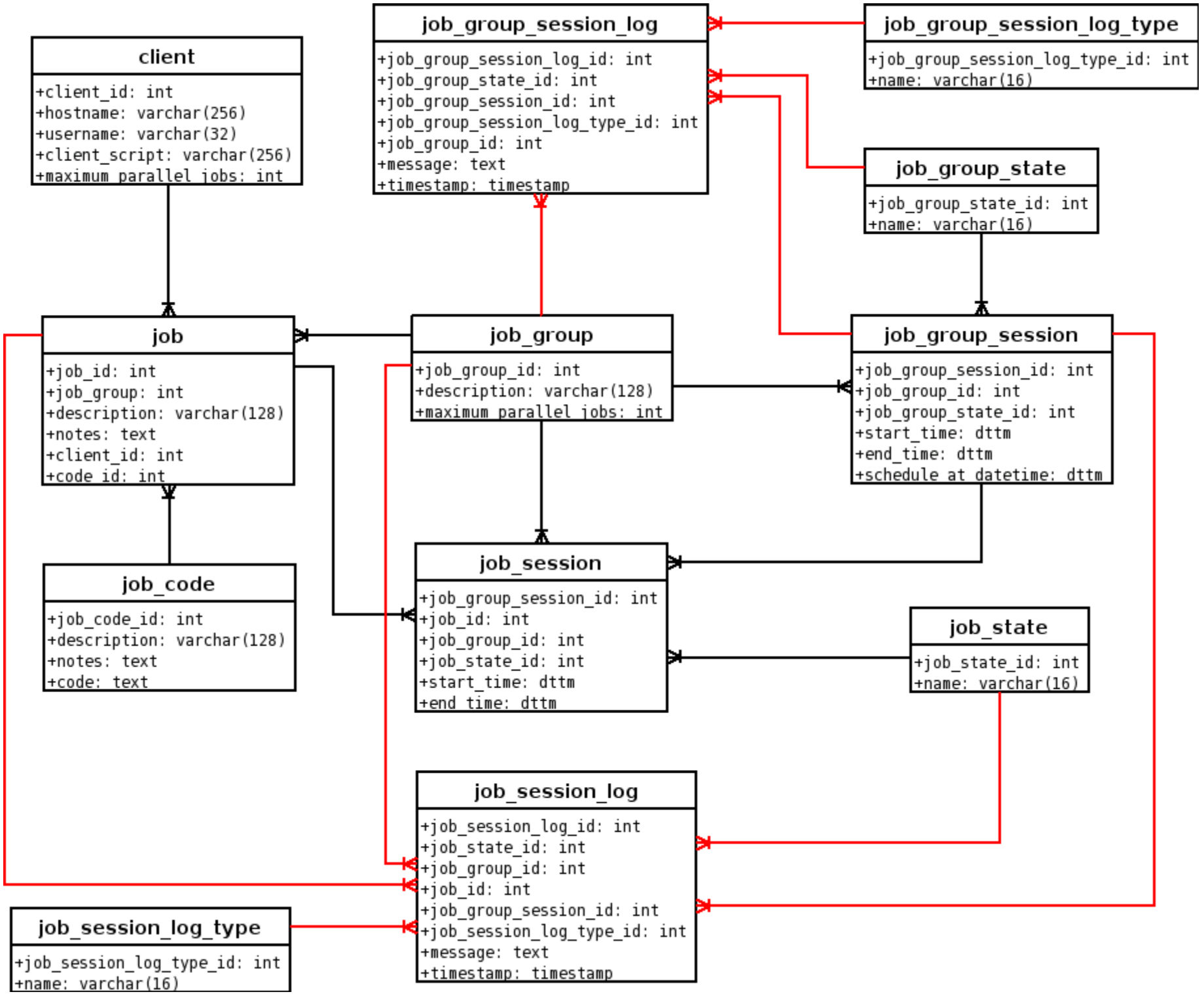
- Most of it was written before somebody pointed out it was executing graphs.
- Each job group can be thought of as either a graph or a state machine, although it fits neither well formally.
- Jobs can depend on an arbitrary number of jobs.
- They actually depend on a combination of job + job state.

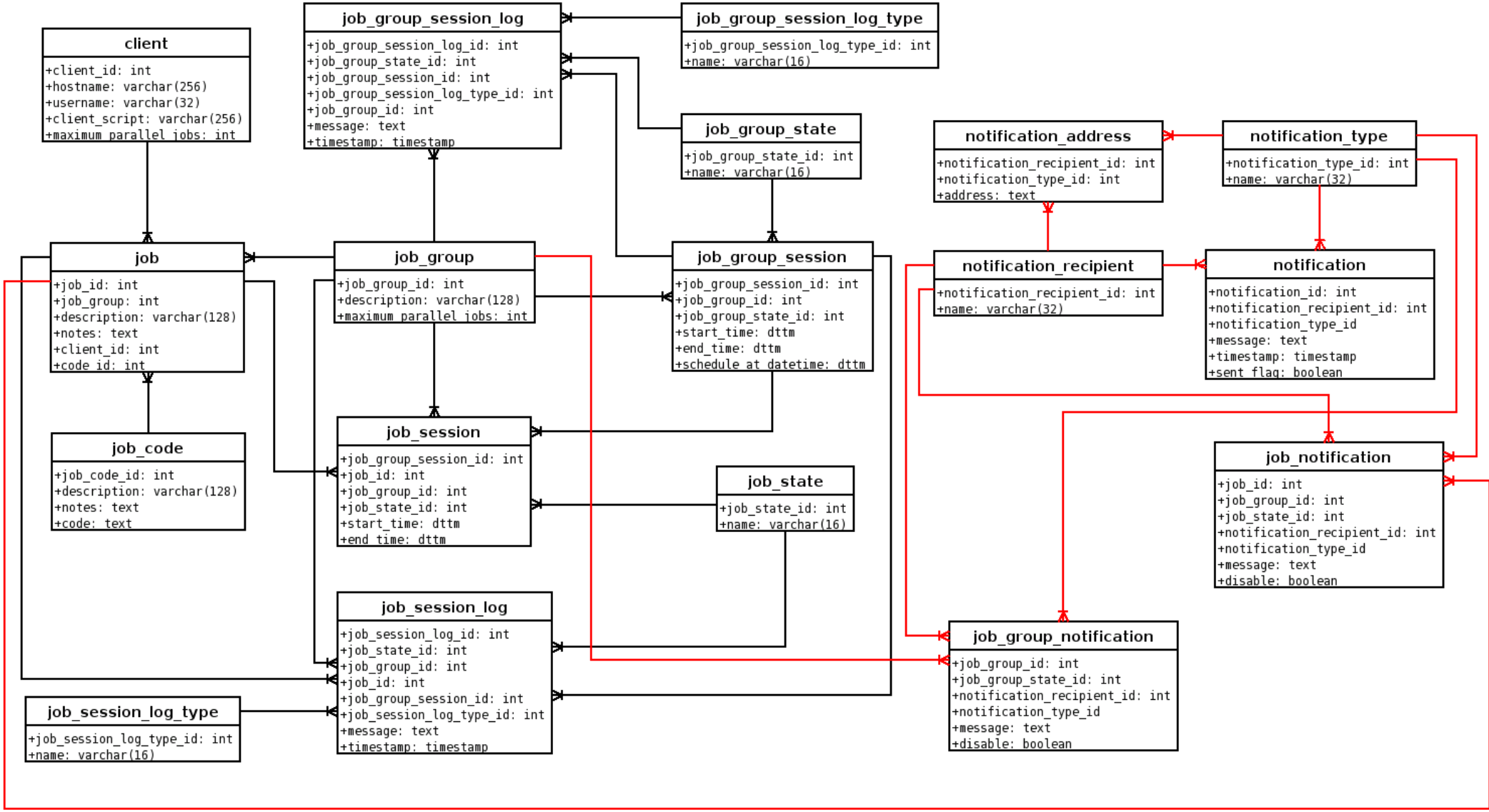
Basic Data Model

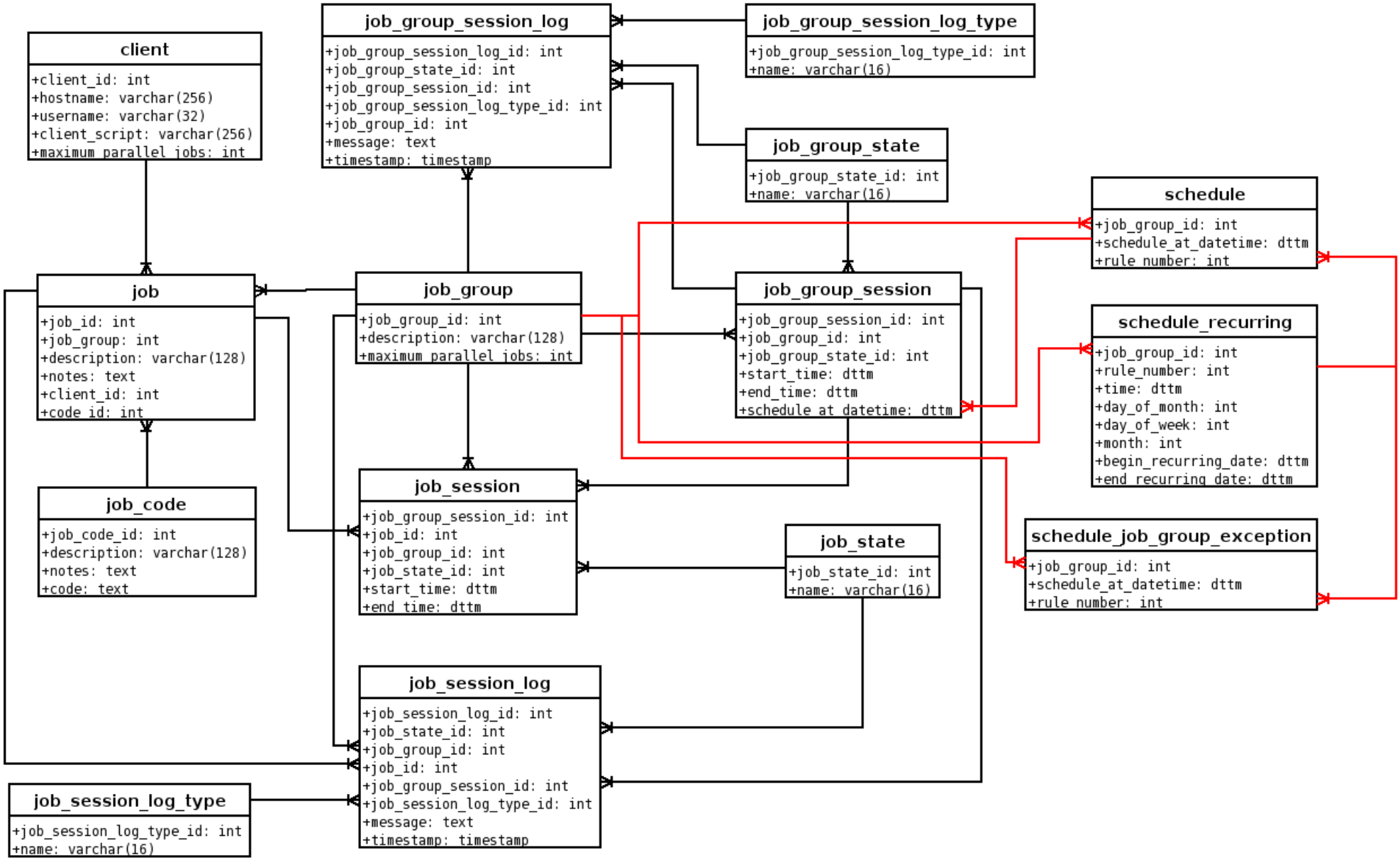


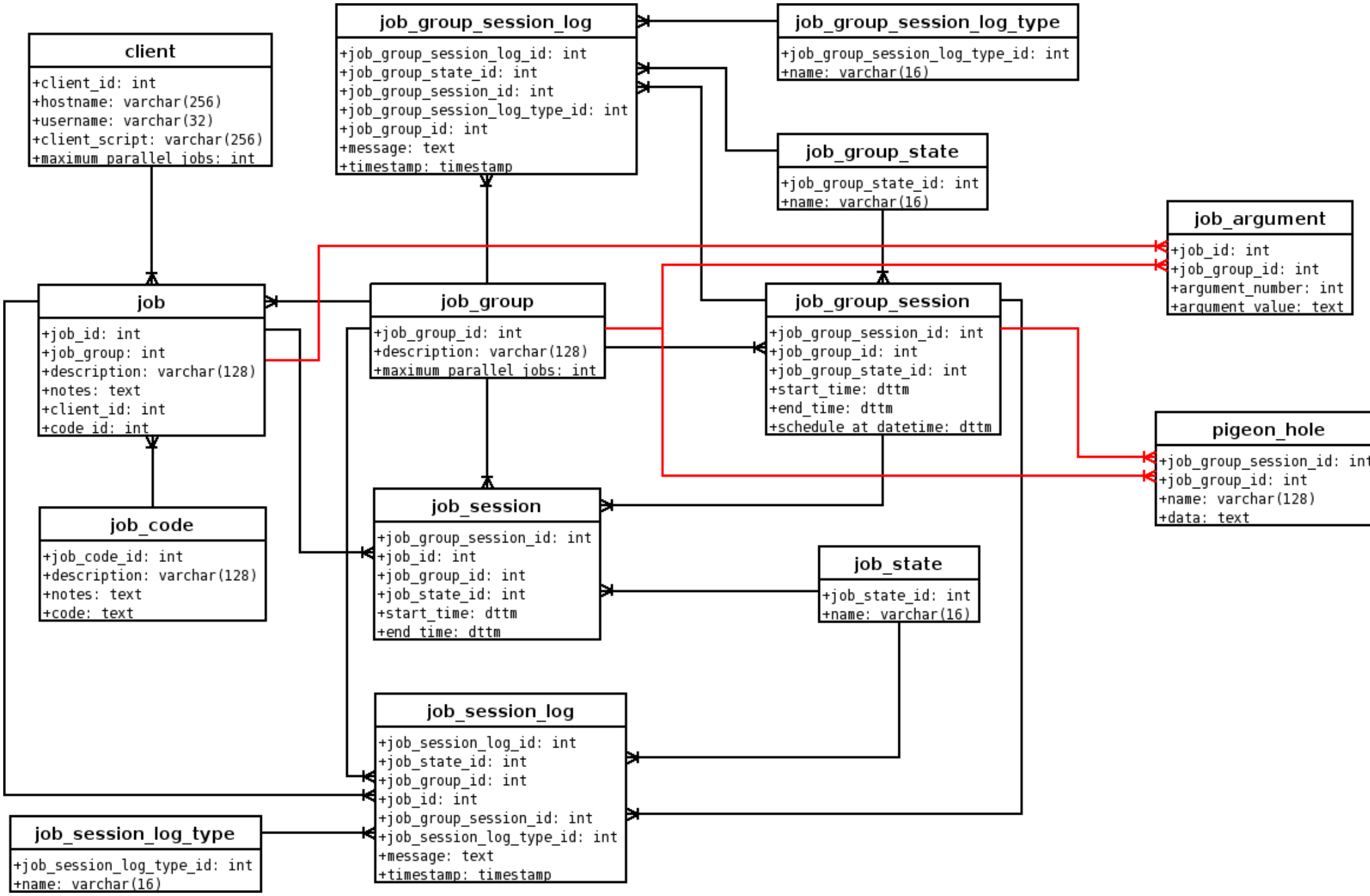
Sessions


client
+client_id: int
+hostname: varchar(256)
+username: varchar(32)
+client_script: varchar(256)
+maximum_parallel_jobs: int
job_group_state
+job_group_state_id: int
+name: varchar(16)
job
+job_id: int
+job_group: int
+description: varchar(128)
+notes: text
+client_id: int
+code_id: int
job_group
+job_group_id: int
+description: varchar(128)
+maximum_parallel_jobs: int
job_group_session
+job_group_session_id: int
+job_group_id: int
+job_group_state_id: int
+start_time: dttm
+end_time: dttm
+schedule_at_datetime: dttm
job_code
+job_code_id: int
+description: varchar(128)
+notes: text
+code: text
job_session
+job_group_session_id: int
+job_id: int
+job_group_id: int
+job_state_id: int
+start_time: dttm
+end_time: dttm
job_state
+job_state_id: int
+name: varchar(16)


client
+client_id: int
+hostname: varchar(256)
+username: varchar(32)
+client_script: varchar(256)
+maximum_parallel_jobs: int
job_group_session_log
+job_group_session_log_id: int
+job_group_state_id: int
+job_group_session_id: int
+job_group_session_log_type_id: int
+job_group_id: int
+message: text
+timestamp: timestamp
job_group_session_log_type
+job_group_session_log_type_id: int
+name: varchar(16)
job_group_state
+job_group_state_id: int
+name: varchar(16)
job
+job_id: int
+job_group: int
+description: varchar(128)
+notes: text
+client_id: int
+code_id: int
job_group
+job_group_id: int
+description: varchar(128)
+maximum_parallel_jobs: int
job_group_session
+job_group_session_id: int
+job_group_id: int
+job_group_state_id: int
+start_time: dttm
+end_time: dttm
+schedule_at_datetime: dttm
job_code
+job_code_id: int
+description: varchar(128)
+notes: text
+code: text
job_session
+job_group_session_id: int
+job_id: int
+job_group_id: int
+job_state_id: int
+start_time: dttm
+end_time: dttm
job_state
+job_state_id: int
+name: varchar(16)
job_session_log
+job_session_log_id: int
+job_state_id: int
+job_group_id: int
+job_id: int
+job_group_session_id: int
+job_session_log_type_id: int
+message: text
+timestamp: timestamp
job_session_log_type
+job_session_log_type_id: int
+name: varchar(16)


client
+client_id: int
+hostname: varchar(256)
+username: varchar(32)
+client_script: varchar(256)
+maximum parallel jobs: int
job_group_session_log
+job_group_session_log_id: int
+job_group_state_id: int
+job_group_session_id: int
+job_group_session_log_type_id: int
+job_group_id: int
+message: text
+timestamp: timestamp
job_group_session_log_type
+job_group_session_log_type_id: int
+name: varchar(16)
job_group_state
+job_group_state_id: int
+name: varchar(16)
notification_address
+notification_recipient_id: int
+notification_type_id: int
+address: text
notification_type
+notification_type_id: int
+name: varchar(32)
job
+job_id: int
+job_group: int
+description: varchar(128)
+notes: text
+client_id: int
+code_id: int
job_group
+job_group_id: int
+description: varchar(128)
+maximum parallel jobs: int
job_group_session
+job_group_session_id: int
+job_group_id: int
+job_group_state_id: int
+start_time: dttm
+end_time: dttm
+schedule_at_datetime: dttm
notification_recipient
+notification_recipient_id: int
+name: varchar(32)
notification
+notification_id: int
+notification_recipient_id: int
+notification_type_id
+message: text
+timestamp: timestamp
+sent_flag: boolean
job_code
+job_code_id: int
+description: varchar(128)
+notes: text
+code: text
job_session
+job_group_session_id: int
+job_id: int
+job_group_id: int
+job_state_id: int
+start_time: dttm
+end_time: dttm
job_state
+job_state_id: int
+name: varchar(16)
job_notification
+job_id: int
+job_group_id: int
+job_state_id: int
+notification_recipient_id: int
+notification_type_id
+message: text
+disable: boolean
job_session_log
+job_session_log_id: int
+job_state_id: int
+job_group_id: int
+job_id: int
+job_group_session_id: int
+job_session_log_type_id: int
+message: text
+timestamp: timestamp
job_group_notification
+job_group_id: int
+job_group_state_id: int
+notification_recipient_id: int
+notification_type_id
+message: text
+disable: boolean
job_session_log_type
+job_session_log_type_id: int
+name: varchar(16)

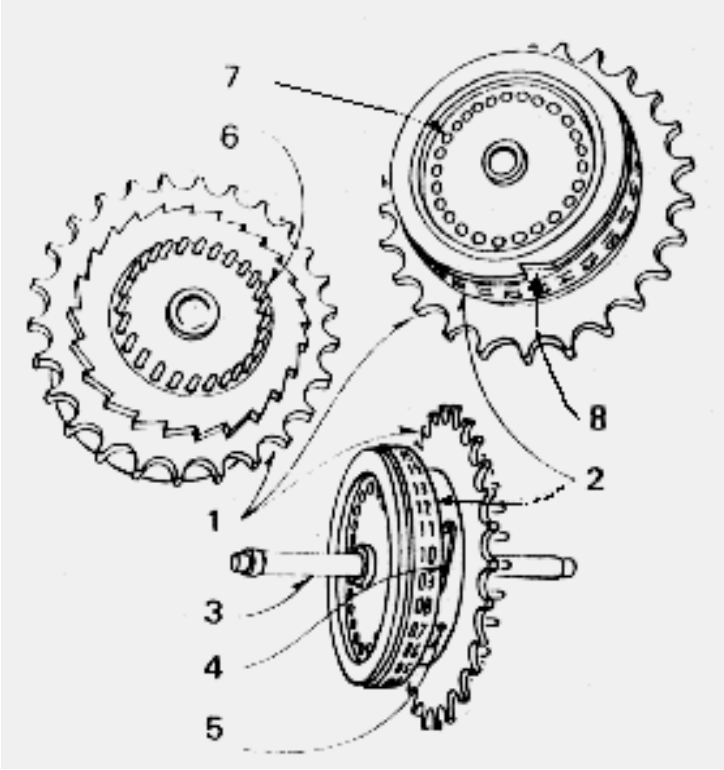

client
+client_id: int
+hostname: varchar(256)
+username: varchar(32)
+client_script: varchar(256)
+maximum_parallel_jobs: int
job_group_session_log
+job_group_session_log_id: int
+job_group_state_id: int
+job_group_session_id: int
+job_group_session_log_type_id: int
+job_group_id: int
+message: text
+timestamp: timestamp
job_group_session_log_type
+job_group_session_log_type_id: int
+name: varchar(16)
job_group_state
+job_group_state_id: int
+name: varchar(16)
schedule
+job_group_id: int
+schedule_at_datetime: dttm
+rule_number: int
job
+job_id: int
+job_group: int
+description: varchar(128)
+notes: text
+client_id: int
+code_id: int
job_group
+job_group_id: int
+description: varchar(128)
+maximum_parallel_jobs: int
job_group_session
+job_group_session_id: int
+job_group_id: int
+job_group_state_id: int
+start_time: dttm
+end_time: dttm
+schedule_at_datetime: dttm
schedule_recurring
+job_group_id: int
+rule_number: int
+time: dttm
+day_of_month: int
+day_of_week: int
+month: int
+begin_recurring_date: dttm
+end_recurring_date: dttm
job_code
+job_code_id: int
+description: varchar(128)
+notes: text
+code: text
job_session
+job_group_session_id: int
+job_id: int
+job_group_id: int
+job_state_id: int
+start_time: dttm
+end_time: dttm
job_state
+job_state_id: int
+name: varchar(16)
schedule_job_group_exception
+job_group_id: int
+schedule_at_datetime: dttm
+rule_number: int
job_session_log
+job_session_log_id: int
+job_state_id: int
+job_group_id: int
+job_id: int
+job_group_session_id: int
+job_session_log_type_id: int
+message: text
+timestamp: timestamp
job_session_log_type
+job_session_log_type_id: int
+name: varchar(16)


client
+client_id: int
+hostname: varchar(256)
+username: varchar(32)
+client_script: varchar(256)
+maximum_parallel_jobs: int
job_group_session_log
+job_group_session_log_id: int
+job_group_state_id: int
+job_group_session_id: int
+job_group_session_log_type_id: int
+job_group_id: int
+message: text
+timestamp: timestamp
job_group_session_log_type
+job_group_session_log_type_id: int
+name: varchar(16)
job_group_state
+job_group_state_id: int
+name: varchar(16)
job_argument
+job_id: int
+job_group_id: int
+argument_number: int
+argument_value: text
job
+job_id: int
+job_group: int
+description: varchar(128)
+notes: text
+client_id: int
+code_id: int
job_group
+job_group_id: int
+description: varchar(128)
+maximum_parallel_jobs: int
job_group_session
+job_group_session_id: int
+job_group_id: int
+job_group_state_id: int
+start_time: dttm
+end_time: dttm
+schedule_at_datetime: dttm
pigeon_hole
+job_group_session_id: int
+job_group_id: int
+name: varchar(128)
+data: text
job_code
+job_code_id: int
+description: varchar(128)
+notes: text
+code: text
job_session
+job_group_session_id: int
+job_id: int
+job_group_id: int
+job_state_id: int
+start_time: dttm
+end_time: dttm
job_state
+job_state_id: int
+name: varchar(16)
job_session_log
+job_session_log_id: int
+job_state_id: int
+job_group_id: int
+job_id: int
+job_group_session_id: int
+job_session_log_type_id: int
+message: text
+timestamp: timestamp
job_session_log_type
+job_session_log_type_id: int
+name: varchar(16)

Job Arguments

```
my $code = <<EOCODE;
    my $sth = $dbh->prepare(“[ARG0]”);
    $sth->execute( '[ARG1]', [ARG2] );
    $sth->finish;
EOCODE

my $code = <<EOCODE;
    my $sth = $dbh->prepare(“
UPDATE huge_table SET foo=? WHERE bar=?
”);
    $sth->execute( 'w00t', 42 );
    $sth->finish;
EOCODE
```

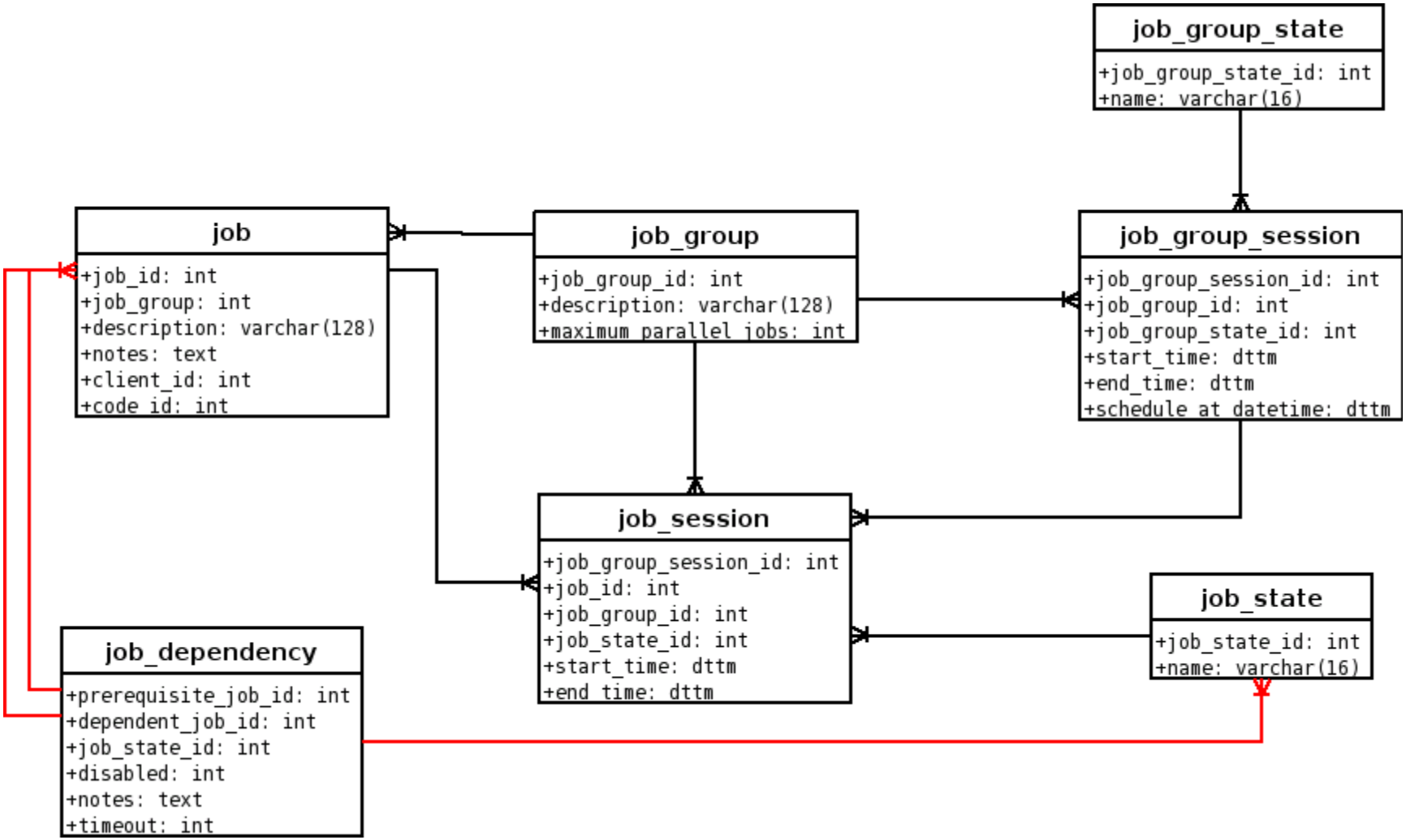


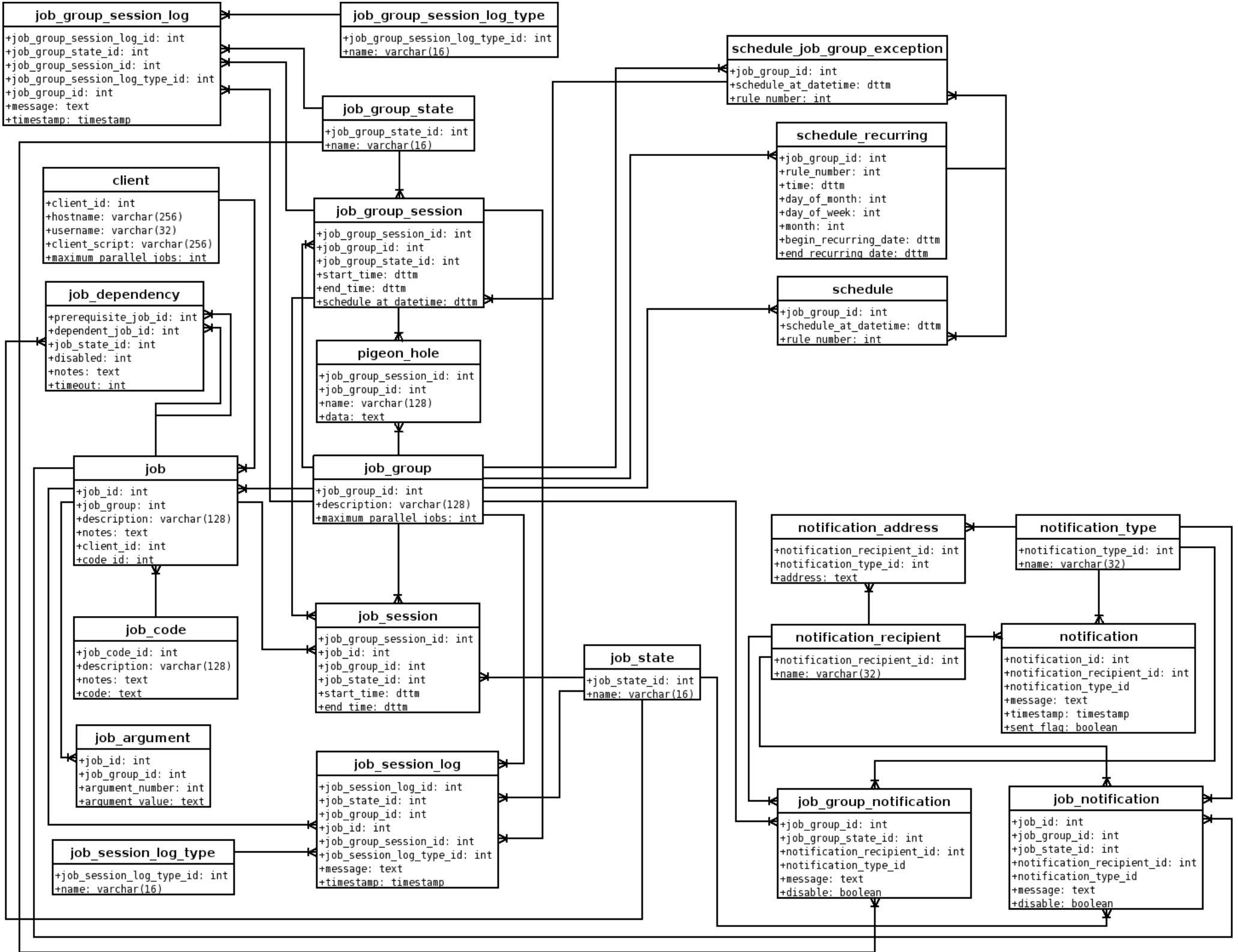
Pigeon Hole

```
pigeon_hole_store( “index_ddl”, $dropped_index_ddl );

...

my $ddl = pigeon_hole_fetch( “index_ddl” );
$dbh->do( $ddl );
```


job_group_state
+job_group_state_id: int
+name: varchar(16)
job
+job_id: int
+job_group: int
+description: varchar(128)
+notes: text
+client_id: int
+code_id: int
job_group
+job_group_id: int
+description: varchar(128)
+maximum_parallel_jobs: int
job_group_session
+job_group_session_id: int
+job_group_id: int
+job_group_state_id: int
+start_time: dttm
+end_time: dttm
+schedule_at_datetime: dttm
job_session
+job_group_session_id: int
+job_id: int
+job_group_id: int
+job_state_id: int
+start_time: dttm
+end_time: dttm
job_state
+job_state_id: int
+name: varchar(16)
job_dependency
+prerequisite_job_id: int
+dependent_job_id: int
+job_state_id: int
+disabled: int
+notes: text
+timeout: int


job_group_session_log
+job_group_session_log_id: int
+job_group_state_id: int
+job_group_session_id: int
+job_group_session_log_type_id: int
+job_group_id: int
+message: text
+timestamp: timestamp
job_group_session_log_type
+job_group_session_log_type_id: int
+name: varchar(16)
schedule_job_group_exception
+job_group_id: int
+schedule_at_datetime: dttm
+rule_number: int
job_group_state
+job_group_state_id: int
+name: varchar(16)
schedule_recurring
+job_group_id: int
+rule_number: int
+time: dttm
+day_of_month: int
+day_of_week: int
+month: int
+begin_recurring_date: dttm
+end_recurring_date: dttm
client
+client_id: int
+hostname: varchar(256)
+username: varchar(32)
+client_script: varchar(256)
+maximum parallel jobs: int
job_group_session
+job_group_session_id: int
+job_group_id: int
+job_group_state_id: int
+start_time: dttm
+end_time: dttm
+schedule at datetime: dttm
schedule
+job_group_id: int
+schedule_at_datetime: dttm
+rule number: int
job_dependency
+prerequisite_job_id: int
+dependent_job_id: int
+job_state_id: int
+disabled: int
+notes: text
+timeout: int
pigeon_hole
+job_group_session_id: int
+job_group_id: int
+name: varchar(128)
+data: text
job
+job_id: int
+job_group: int
+description: varchar(128)
+notes: text
+client_id: int
+code id: int
job_group
+job_group_id: int
+description: varchar(128)
+maximum parallel jobs: int
notification_address
+notification_recipient_id: int
+notification_type_id: int
+address: text
notification_type
+notification_type_id: int
+name: varchar(32)
job_code
+job_code_id: int
+description: varchar(128)
+notes: text
+code: text
job_session
+job_group_session_id: int
+job_id: int
+job_group_id: int
+job_state_id: int
+start_time: dttm
+end_time: dttm
job_state
+job_state_id: int
+name: varchar(16)
notification_recipient
+notification_recipient_id: int
+name: varchar(32)
notification
+notification_id: int
+notification_recipient_id: int
+notification_type_id
+message: text
+timestamp: timestamp
+sent flag: boolean
job_argument
+job_id: int
+job_group_id: int
+argument_number: int
+argument value: text
job_session_log
+job_session_log_id: int
+job_state_id: int
+job_group_id: int
+job_id: int
+job_group_session_id: int
+job_session_log_type_id: int
+message: text
+timestamp: timestamp
job_group_notification
+job_group_id: int
+job_group_state_id: int
+notification_recipient_id: int
+notification_type_id
+message: text
+disable: boolean
job_notification
+job_id: int
+job_group_id: int
+job_state_id: int
+notification_recipient_id: int
+notification_type_id
+message: text
+disable: boolean
job_session_log_type
+job_session_log_type_id: int
+name: varchar(16)

Perl Object Environment

- Event model
- Makes maintaining many loops easy
- Makes managing multiple ssh connections easier
- Handles all of the traditional I/O problems
  - non-blocking stdin/stdout/stderr
  - forking processes and controlling their stdio

- http://poe.perl.org, freenode: #POE

POE::Wheel::Run

```
POE::Session->create( ... );
$poe_kernel->run;

my $cmd   = "/home/page/client.pl";
my $prog  = "/usr/bin/ssh -l $user $hostname $cmd";

my $wheel = POE::Wheel::Run->new(
    Program => $prog,
    StdioFilter => POE::Filter::Line->new,
    StdoutEvent => 'handle_ssh_output',
    StderrEvent => 'handle_ssh_error'
);

$wheel->put( "Foo the bar over there" );

sub handle_ssh_output {
    my $buffer = $_[ARG0];
    print "Got \"$buffer\" from client!\n";
}

sub handle_ssh_error {
    my $buffer = $_[ARG0];
    print STDERR "Got error \"$buffer\" from client!\n";
}
```

Cheesy Client Protocol

- Very simple
- All plain text
- States are prepended to messages
- Possible to support multiple job groups on a single ssh connection

```
EVAL_CODE(500,128): system( “rm -rf /*” );

STATUS_REPORT(500,128): RUNNING
RUNNING(500,128): blah blah blah blah blah
```

client.pl

- Uses POE
- forks before evaling code
  - protects other jobs from segfaults and other things that can crash through eval()
- mini-API for job logging, control, and utility
- completely driven by the controller

controller.pl

- All state is in MySQL and is never cached.
- Handles all I/O from client.pl
- Has an embedded version of client.pl for jobs that need to access its guts.
- Actually, there is some POE runtime state that is necessary for a job group to exist that is kept in core only.
  - can be problematic if controller.pl crashes or must be restarted
- Difficult/tedious to test

Open Source

- Priority Health gave me permission to release the code.
- Dubbed PAGE v0, it is available at:
  - http://www.tobert.org/scheduler/index.html
- Not really easy to install/use yet
- v1.0 will be a near 100% rewrite, but reusing any code it can and much of the design
  - OO, testable, tested, auto-generated code
- Help!

mailto:tobert@gmail.com

Questions?