A Primer for the pfx_dw Data Warehouse Schema

The **dwh** schema is a fairly textbook star schema, with things like jobs and worker usage in **fact** tables, and the various things that you might use in a SQL `WHERE` clause in **dimension** tables.

The job_fact table, a typical "fact" table.

For example, to get reports about various jobs over time, you'll be querying the **dwh.job_fact** table:

```
pfx=> \dS+ dwh.job_fact
                                    Table "dwh.job_fact"
      Column      |           Type           | Modifiers | Storage  | Stats target |
Description
------------------+--------------------------+-----------+----------+--------------+--
-----------
 time_sk          | integer                  | not null  | plain    |              |
 job_id           | integer                  |           | plain    |              |
 job_pgrp         | integer                  |           | plain    |              |
 job_name         | character varying(255)   |           | extended |              |
 job_priority     | integer                  |           | plain    |              |
 job_timesubmit   | timestamp with time zone |           | plain    |              |
 job_timestart    | timestamp with time zone |           | plain    |              |
 job_timecomplete | timestamp with time zone |           | plain    |              |
 jobstatus_sk     | integer                  |           | plain    |              |
 user_sk          | smallint                 |           | plain    |              |
 cluster_sk       | smallint                 |           | plain    |              |
 account_sk       | smallint                 |           | plain    |              |
 kind_sk          | smallint                 |           | plain    |              |
 prototype_sk     | smallint                 |           | plain    |              |
 prod_show_sk     | smallint                 |           | plain    |              |
 prod_shot_sk     | smallint                 |           | plain    |              |
 prod_seq_sk      | smallint                 |           | plain    |              |
 prod_client_sk   | smallint                 |           | plain    |              |
 prod_dept_sk     | smallint                 |           | plain    |              |
 prod_custom1_sk  | smallint                 |           | plain    |              |
 prod_custom2_sk  | smallint                 |           | plain    |              |
 prod_custom3_sk  | smallint                 |           | plain    |              |
 prod_custom4_sk  | smallint                 |           | plain    |              |
 prod_custom5_sk  | smallint                 |           | plain    |              |
 cpu_count        | integer                  | not null  | plain    |              |
 cpu_seconds      | integer                  | not null  | plain    |              |
 work_count       | integer                  | not null  | plain    |              |
 worktime_min     | integer                  | not null  | plain    |              |
 worktime_max     | integer                  | not null  | plain    |              |
 worktime_avg     | real                     | not null  | plain    |              |
Indexes:
    "job_id_unique" UNIQUE CONSTRAINT, btree (job_id)
    "idx_time" btree (time_sk)
```

⚠ **job_time* are stored as timestamp types**
With Qube 7.x which uses PostgreSQL as the backbone, job_time* are stored as timestamp types, so, for example, you can fetch time data for a specific job as in:

```
pfx=> SELECT job_id, job_timesubmit, job_timestart,job_timecomplete FROM dwh.job_fact WHERE
job_id=57 ;
 job_id |     job_timesubmit     |     job_timestart      |    job_timecomplete
--------+------------------------+------------------------+------------------------
     57 | 2020-08-07 23:49:57-10 | 2020-08-07 23:49:57-10 | 2020-08-07 23:50:03-10
(1 row)
```

"*_sk" columns can be used to do INNER JOINs to a similarly named dimension table

Any column that is named with an **_sk** suffix is a **S**ynthetic **K**ey that points to a corresponding dimension table, named with the part of the column before the **_sk**; the dimension table will have a **_dim** suffix in the name. This way, it's easy to write the **JOIN**'s, the column name is a clue to the dimension table, which will have a column of the same name. Almost every dimension table will consist of a ***_sk** `PRIMARY KEY` and a **name** co lumn.

A typical dimension table, the "user_dim" table

For example, the **user_sk** column can be used to do a SQL `INNER JOIN` to the **user_dim** table.

```
pfx=> \dS+ dwh.user_dim
                                                    Table "dwh.user_dim"
 Column  |          Type          |                       Modifiers
| Storage  | Stats target | Description
---------+------------------------+-------------------------------------------------------
---------+----------+--------------+-------------
 user_sk | integer                | not null default
nextval('user_dim_user_sk_seq'::regclass) | plain    |              |
 name    | character varying(255) | not null
| extended |              |
Indexes:
    "user_dim_pkey" PRIMARY KEY, btree (user_sk)
    "user_dim_name_key" UNIQUE CONSTRAINT, btree (name)

pfx=> SELECT name FROM user_dim;
  name
--------
 shinya
 root
 administrator
 jburk
 joe
 kmac
(6 rows)
```

Get a count of all jobs for a particular user:

```
pfx=> SELECT COUNT(*) FROM job_fact INNER JOIN user_dim ON job_fact.user_sk =
user_dim.user_sk WHERE user_dim.name = 'shinya';
 count
-------
   370
(1 row)
```

The time dimension table

The **dwh.time_dim** table is provided so that you don't have to perform date/time operations on every row in a **fact** table (since they can run into the 100's of millions of rows), instead you do a SQL `INNER JOIN` to it and use the values in the **time_dim** table in your `WHERE` clause. The **time_sk** column in every fact table has an identical value in the `time_dim` table which has a single row with a primary key time_sk. The `time_sk` value is actually the unix epoch time in seconds:

```
pfx=> SELECT * FROM time_dim ORDER BY time_sk DESC LIMIT 1;
  time_sk   |       date_time        | hour |    date    | dow | month_name | month |
year
------------+------------------------+------+------------+-----+------------+-------+-
-----
 1607570700 | 2020-12-09 17:25:00-10 |   17 | 2020-12-09 |   3 | December   |    12 |
2020
(1 row)

pfx=> SELECT to_timestamp(time_sk), date_time FROM time_dim ORDER BY time_sk DESC
LIMIT 1;
      to_timestamp      |       date_time
------------------------+------------------------
 2020-12-09 17:30:00-10 | 2020-12-09 17:30:00-10
(1 row)
```

The "job status" dimension table

The **dwh.jobstatus_dim** table is one of the few exceptions to the normal dimension table structure; it provides a mapping between the integer and human-readable status values.

```
pfx=> SELECT * FROM jobstatus_dim;
 jobstatus_sk | status_int | status_char | effective_date | expiry_date
--------------+------------+-------------+----------------+-------------
            1 |         16 | complete    | 1999-12-31     | 9999-12-31
            2 |         32 | failed      | 1999-12-31     | 9999-12-31
            3 |         48 | killed      | 1999-12-31     | 9999-12-31
            4 |        261 | dying       | 1999-12-31     | 9999-12-31
            5 |        262 | exiting     | 1999-12-31     | 9999-12-31
            6 |        265 | registering | 1999-12-31     | 9999-12-31
            7 |        272 | blocked     | 1999-12-31     | 9999-12-31
            8 |        288 | waiting     | 1999-12-31     | 9999-12-31
            9 |        304 | suspended   | 1999-12-31     | 9999-12-31
           10 |        320 | pending     | 1999-12-31     | 9999-12-31
           11 |        323 | waiting     | 1999-12-31     | 9999-12-31
           12 |        325 | badlogin    | 1999-12-31     | 9999-12-31
           13 |        336 | running     | 1999-12-31     | 9999-12-31
(13 rows)
```

Get a count of all jobs for a particular user for August, 2020:

```
pfx=> SELECT COUNT(*) FROM job_fact INNER JOIN user_dim ON job_fact.user_sk =
user_dim.user_sk INNER JOIN time_dim ON job_fact.time_sk=time_dim.time_sk WHERE
user_dim.name = 'shinya' AND time_dim.month = 8 AND time_dim.year = 2020;
 count
-------
    96
(1 row)
```

Get a count of all jobs for each user for all of 2020:

```
pfx=> SELECT user_dim.name,time_dim.month_name,COUNT(*) as "job count" FROM job_fact
INNER JOIN user_dim ON job_fact.user_sk = user_dim.user_sk INNER JOIN time_dim ON
job_fact.time_sk=time_dim.time_sk WHERE time_dim.year = 2020 GROUP BY
user_dim.name,time_dim.month,time_dim.month_name ORDER BY
user_dim.name,time_dim.month;
  name  | month_name | job count
--------+------------+-----------
 root   | July       |         2
 root   | August     |         3
 root   | September  |        13
 root   | October    |         8
 root   | November   |         4
 shinya | July       |         9
 shinya | August     |        31
 shinya | September  |       157
 shinya | October    |        50
 shinya | November   |        75
(10 rows)
```

Get a count of all jobs for each user for all of 2020, broken down by month and the job's final status:

```
pfx=> SELECT
  user_dim.name
  , time_dim.year
  , time_dim.month_name
  , jobstatus_dim.status_char
  , COUNT(*) as "job count"
FROM
  job_fact
INNER JOIN
  user_dim
ON
  job_fact.user_sk=user_dim.user_sk
INNER JOIN
  time_dim
ON
  job_fact.time_sk=time_dim.time_sk
INNER JOIN
  jobstatus_dim
ON
  job_fact.jobstatus_sk=jobstatus_dim.jobstatus_sk
WHERE
  time_dim.year = 2020
GROUP BY
  user_dim.name,time_dim.year,time_dim.month_name,jobstatus_dim.status_char
  , time_dim.month
  , jobstatus_dim.status_int
ORDER BY
  user_dim.name
  , time_dim.month
  ,jobstatus_dim.status_char
;
  name  | year | month_name | status_char | job count
--------+------+------------+-------------+-----------
 root   | 2020 | July       | complete    |         2
 root   | 2020 | August     | complete    |         3
 root   | 2020 | September  | complete    |        11
 root   | 2020 | September  | failed      |         2
 root   | 2020 | October    | complete    |         8
 root   | 2020 | November   | complete    |         4
 shinya | 2020 | July       | complete    |         6
 shinya | 2020 | July       | failed      |         1
 shinya | 2020 | July       | killed      |         2
 shinya | 2020 | August     | complete    |        25
 shinya | 2020 | August     | killed      |         6
 shinya | 2020 | September  | complete    |       139
 shinya | 2020 | September  | failed      |        11
 shinya | 2020 | September  | killed      |         7
 shinya | 2020 | October    | complete    |        47
 shinya | 2020 | October    | failed      |         2
 shinya | 2020 | October    | killed      |         1
 shinya | 2020 | November   | complete    |        64
 shinya | 2020 | November   | failed      |         8
 shinya | 2020 | November   | killed      |         3
(20 rows)
```

Get the sum total of cpu_seconds used for each user for the last 7 days, broken down by user, date, and the job's final status:

```
pfx=> SELECT
    user_dim.name
    , time_dim.date
    , jobstatus_dim.status_char
    , SUM(job_fact.cpu_seconds) as "cpu_time"
FROM
    job_fact
INNER JOIN
    user_dim
ON
    job_fact.user_sk=user_dim.user_sk
INNER JOIN
    time_dim
ON
    job_fact.time_sk=time_dim.time_sk
INNER JOIN
    jobstatus_dim
ON
    job_fact.jobstatus_sk=jobstatus_dim.jobstatus_sk
WHERE
    DATE_PART('day', CURRENT_DATE - time_dim.date_time) < 7
GROUP BY
    user_dim.name,jobstatus_dim.status_char
    , time_dim.date
    , jobstatus_dim.status_int
ORDER BY
    time_dim.date
    , cpu_time DESC
    , jobstatus_dim.status_char
;
  name  |    date    | status_char | cpu_time
--------+------------+-------------+----------
 shinya | 2020-11-13 | complete    |      144
 shinya | 2020-11-16 | complete    |       97
 shinya | 2020-11-17 | failed      |      906
 shinya | 2020-11-17 | killed      |      677
 shinya | 2020-11-17 | complete    |      102
 shinya | 2020-11-18 | failed      |    18695
 root   | 2020-11-18 | complete    |     1199
 shinya | 2020-11-18 | complete    |      760
 shinya | 2020-11-19 | failed      |     8022
 shinya | 2020-11-19 | complete    |      606
 root   | 2020-11-19 | complete    |       46
 shinya | 2020-11-19 | killed      |        0
(12 rows)
```