Sandpiper (/spaces/127/sandpiper) ‣ Discussions (...

Private Space · Technology Standards (/spaces/9/technology-standards/technologyhome)

(/spaces/127/sandpiper)

Activity Stream (/spaces/127/sandpiper/?act=1)      People (/spaces/127/sandpiper/people)      Content ▾

**Posted in:** API (/spaces/127/sandpiper/forums/5044/api)

# Data Model (ER Diagram)

✉ Unsubscribe from this discussion

🔊 Subscribe to RSS (../../../../../spaces-cf/forums/rss-space-posts.ashx?
spaceID=127&topicID=5271&forumID=5044&key=rrer%2B1xDo%2BlxpC7jBRbM5w%3D%3D)

**Doug Winsby (https://autocare.communifire.com/people/dougwinsby)** 🔶🔶🔶🔶🔶     ⋮
10/18/2019

We're using dbdiagram.io (https://dbdiagram.io/home) to model the database because:

1. It's free!

2. It handles the (one-page) Sandpiper data model nicely

3. It uses a text-based definition language that can be checked into GitHub

4. It makes pretty (one-page) diagrams

5. It generates PostgreSQL create scripts

Here is a read-only link to the Sandpiper dbdiagram project:

https://dbdiagram.io/d/5d72619383427516dc0b67c2
(https://dbdiagram.io/d/5d72619383427516dc0b67c2)

er-diagram (/spaces/127/sandpiper/searchresults?keyword=%22er-diagram%22&searchtags=1)

👍 Like

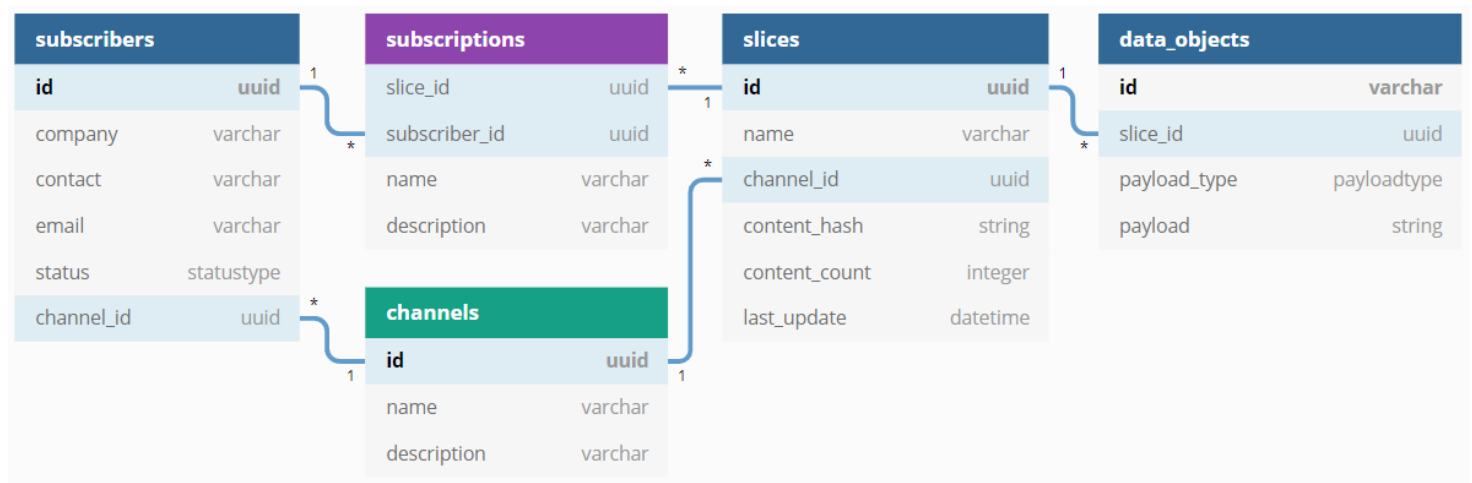Reply (/forums/post?tid=5271&ReplyPostID=5272&SpaceID=127)

**Doug Winsby (https://autocare.communifire.com/people/dougwinsby)** 🔶🔶🔶🔶🔶     ⋮

One requirement we've been underlined discussing (https://autocare.communifire.com/spaces/127/sandpiper/forums/api/5242/pools-vs-slices?postid=5257#5257) is the need to relate "subscribers" to "slices". This seems like a pretty simple relationship, but I see two different approaches when modelling it:

| subscribers | |
|---|---|
| **id** | uuid |
| company | varchar |
| contact | varchar |
| email | varchar |
| status | statustype |
| channel_id | uuid |

| subscriptions | |
|---|---|
| slice_id | uuid |
| subscriber_id | uuid |
| name | varchar |
| description | varchar |

| channels | |
|---|---|
| **id** | uuid |
| name | varchar |
| description | varchar |

| slices | |
|---|---|
| **id** | uuid |
| name | varchar |
| channel_id | uuid |
| content_hash | string |
| content_count | integer |
| last_update | datetime |

| data_objects | |
|---|---|
| **id** | varchar |
| slice_id | uuid |
| payload_type | payloadtype |
| payload | string |

**Purple Option.** This approach models a simple many-to-many relationship using a "subscriptions" join table. This would allow any subscriber to be "assigned" any slice, and any slice to be assigned to any subscriber. It's the "wild west" option.

**Green Option**. This approach models a much more restrictive relationship between subscribers and slices. In this case, a slice is assigned to **only one** "channel", but each channel can be assigned to multiple subscribers. Also, a subscriber is assigned to just one channel.

In some ways, the "Green Option" seems more in-line with our thinking about `slices` being **specific to a customer relationship** (e.g. "AAP" `channel` includes "AAP Brake Friction", "AAP Brake Hardware" `slices` ). But I don' want to limit a valid use case where we might want to share slices with anyone.

Any thoughts?

slices-to-subscribers (/spaces/127/sandpiper/searchresults?keyword=%22slices-to-subscribers%22&searchtags=1)

👍 Like

Reply (/forums/post?tid=5271&ReplyPostID=5273&SpaceID=127)          Answer