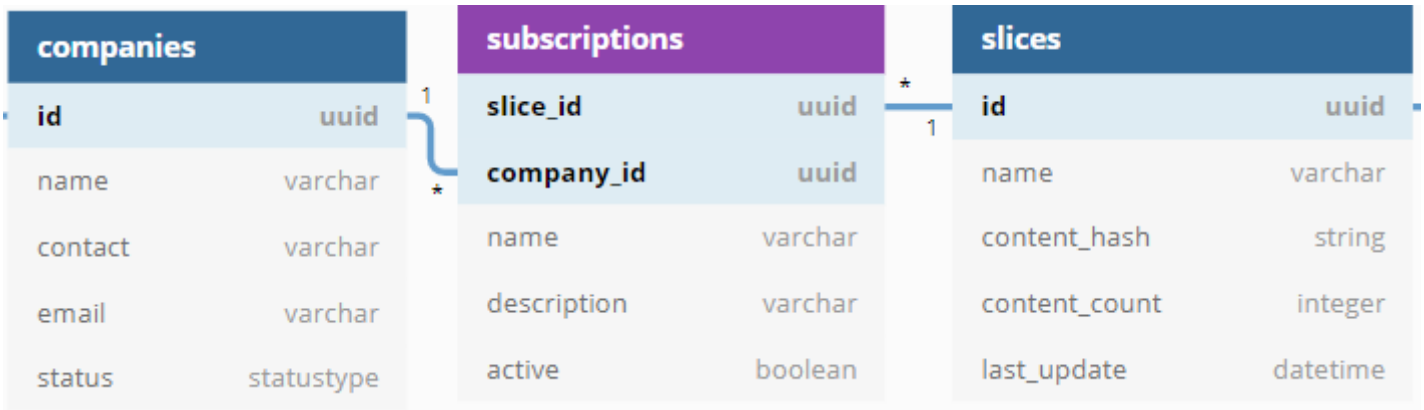
Posted in: Development (/spaces/127/sandpiper/forums/5366/development)

Subscription Table

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
1/14/2020

Here is the current ER diagram showing the "subscription" relationship.



This is a classic many-to-many relationship between `companies` and `slices`. But we've decided to add a primary key "id" to this "junction" table because of this advise:

- Consider using a separate ID field for the junction/join table if it contains fields other than the ID. This tends to note that it is a **first class entity**.
- Consider using a separate ID field if APIs or any existing logic tend to use single fields for retrieving/editing entities. That can help other people follow your code in the context of a larger project.

Both of these points hold true for this table.

We plan on having API resource end-points like the following for listing subscriptions (without the "id" required):

```
/companies/{comp-id}/subscriptions
/companies/{comp-id}/subscriptions/{slice-id}
```

But we'd also like to be able to update the subscription "name", for example, with a simple:

```
/subscriptions/{id}
```

Because these subscriptions will not be synced, they can just use a simple integer (serial) primary key.

junction-table (/spaces/127/sandpiper/searchresults?keyword=%22junction-table%22&searchtags=1)
subscriptions (/spaces/127/sandpiper/searchresults?keyword=%22subscriptions%22&searchtags=1)

Like

Reply (/forums/post?tid=5538&ReplyPostID=5539&SpaceID=127)

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
1/17/2020

Here is an example response from a `GET /subscriptions/1` request (notice that it also shows the related company and slice information):

```
{
  "id": 1,
  "name": "Subscription 1 (company1, slice2)",
  "description": "",
  "active": true,
  "created_at": "2020-01-17T03:30:05.656288Z",
  "updated_at": "2020-01-17T03:30:05.656288Z",
  "company": {
    "id": "20000000-0000-0000-0000-000000000000",
    "name": "Retailer",
    "sync_addr": "https://sandpiper.retailer.com",
    "active": true,
    "created_at": "2020-01-16T23:50:29.961345Z",
    "updated_at": "2020-01-16T23:50:29.961345Z"
  },
  "slice": {
    "id": "2bea8308-1840-4802-ad38-72b53e31594c",
    "slice_name": "Slice2",
    "content_hash": "2268e5deabf5e6d0740cd1a77df56f67093ec943",
    "content_count": 1,
    "content_date": "0001-01-01T00:00:00Z",
    "created_at": "2020-01-17T03:29:38.974349Z",
    "updated_at": "2020-01-17T03:29:38.974349Z"
  }
}
```


Like

Reply (/forums/post?tid=5538&ReplyPostID=5556&SpaceID=127) Answer

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

8/6/2020

It turns out that we do want to sync subscriptions, and so we made the identifier a UUID.

```
CREATE TABLE IF NOT EXISTS "subscriptions" (
  "sub_id"       uuid PRIMARY KEY,
  "slice_id"     uuid REFERENCES "slices" ("id") ON DELETE RESTRICT,
  "company_id"   uuid REFERENCES "companies" ("id") ON DELETE RESTRICT,
  "name"         text NOT NULL,
  "description"  text,
  "active"       boolean,
  "created_at"   timestamp,
  "updated_at"   timestamp,
  CONSTRAINT "sub_alt_key" UNIQUE("slice_id", "company_id")
);
CREATE UNIQUE INDEX ON subscriptions (lower(name));
```

We used "sub_id" instead of just "id" due to a problem with the ORM (or more likely, not fully understanding how to use the ORM).

Like

Reply (/forums/post?tid=5538&ReplyPostID=6091&SpaceID=127) Answer

Adrian Parker (https://autocare.communifire.com/people/aparker)
8/10/2020

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

Which ORM are you using?

Like

Reply (/forums/post?tid=5538&ReplyPostID=6092&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
8/10/2020

https://pg.uptrace.dev/ (https://pg.uptrace.dev/)

Not really a traditional ORM, but it does help avoid SQL injection (and is somewhat prettier than raw SQL).

Example 1:

```
err := db.Model(slice).Relation("Companies", filterFn).WherePK().Select()
```

Example 2:

```
err := db.Model(&tagged).Column("slice.id").
 Join("INNER JOIN slice_tags AS st ON slice.id = st.slice_id").
 Join("INNER JOIN tags AS t ON st.tag_id = t.id").
 Where("t.name IN (?)", pg.In(tags.TagList)).
 Group("slice.id").
 Having("COUNT(slice.id) = ?", tags.Count()).Select()
```

orm (/spaces/127/sandpiper/searchresults?keyword=%22orm%22&searchtags=1)

Like

Reply (/forums/post?tid=5538&ReplyPostID=6093&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
8/10/2020

The "models" (structs) are where you see the ORM relationships:

```
type Company struct {
 ID            uuid.UUID       `json:"id"`
 Name          string          `json:"name"`
 SyncAddr      string          `json:"sync_addr"`
 SyncAPIKey    string          `json:"sync_api_key,omitempty"` // only on secondar
y
 SyncUserID    int             `json:"sync_user_id,omitempty"` // only on primary
 Active        bool            `json:"active"`
 CreatedAt     time.Time       `json:"created_at"`
 UpdatedAt     time.Time       `json:"updated_at"`
 Users         []*User         `json:"users,omitempty"`         // has-many relati
on
 SyncUser      *User           `json:"sync_user,omitempty"`     // has-one relatio
n
 Subscriptions []*Subscription `json:"subscriptions,omitempty"` // has-many relati
on
}
```

Like

Reply (/forums/post?tid=5538&ReplyPostID=6094&SpaceID=127) Answer