Interactive - 26 - 1 to 1 relationship (pk to pk)

This exercise creates a bunch of tables, triggers indexes. Run each of the files.

The application uses a small set of user authorization tables from a wrapper application. We have to create the tables and the tables that match with in the application.

Our 1 to 1 relationship is between `"t_ymux_user"` and our application table `ct_login`. ct_login privies extra application specific columns that every user has to have.

We are just going to create all 6 tables - then explore the relationships.

3 Tables From The Security Application

The Tables are “t_ymux_user” , “ct_homework_ans” and “t_ymux_user_log”
.

Run the file `hw26_01.sql`

```
CREATE EXTENSION if not exists "uuid-ossp";
CREATE EXTENSION if not exists pgcrypto;

DROP TABLE if exists "t_ymux_user" cascade ;

CREATE TABLE "t_ymux_user" (
      "id"                    uuid DEFAULT uuid_generate_v4() not null primary key
    , "username"          text
    , "password"          text
    , "realm"               text
    , "real_name"           text
    , "salt"              text
    , "email"               text
    , "email_confirmed"   char varying (1) default 'n' not null
    , "setup_2fa_complete"    char varying (1) default 'n' not null
    , "rfc_6238_secret"       text
    , "recovery_token"        text
    , "recovery_expire"   timestamp
    , "parent_user_id"        uuid
    , "org_user_id"           uuid
    , "auth_token"            text
    , "acct_expire"       timestamp
    , "updated"           timestamp
    , "created"           timestamp default current_timestamp not null
);
```

```
create unique index "t_ymux_user_u1" on "t_ymux_user" ( "username" );
create index "t_ymux_user_p1" on "t_ymux_user" ( "email" );
create unique index "t_ymux_user_u3" on "t_ymux_user" ( "recovery_token" );
create index "t_ymux_user_p2" on "t_ymux_user" ( "created", "setup_2fa_complete" );
create index "t_ymux_user_p3" on "t_ymux_user" ( "created", "email_confirmed" );



CREATE OR REPLACE function t_ymux_user_upd()
RETURNS trigger AS $$
BEGIN
    NEW.updated := current_timestamp;
    RETURN NEW;
END
$$ LANGUAGE 'plpgsql';


CREATE TRIGGER t_ymux_user_trig
    BEFORE update ON "t_ymux_user"
    FOR EACH ROW
    EXECUTE PROCEDURE t_ymux_user_upd();
```

Run the file `hw26_02.sql`

```
-- ------------------------------------------------------------ -- ----------------------------
-- Note the "auth_token" is the "ID" for this row. (Primnary Key)
-- ------------------------------------------------------------ -- ----------------------------

drop TABLE if exists "t_ymux_auth_token" cascade ;

CREATE TABLE "t_ymux_auth_token" (
      "id"                    uuid DEFAULT uuid_generate_v4() not null primary key
    , "user_id"               uuid not null
    , "updated"           timestamp
    , "created"           timestamp default current_timestamp not null
);

create index "t_ymux_auth_token_p1" on "t_ymux_auth_token" ( "user_id" );
create index "t_ymux_auth_token_p2" on "t_ymux_auth_token" ( "created" );


ALTER TABLE "t_ymux_auth_token"
    ADD CONSTRAINT "t_ymux_auth_token_user_id_fk1"
    FOREIGN KEY ("user_id")
    REFERENCES "t_ymux_user" ("id")
```

```
;

CREATE OR REPLACE function t_ymux_auth_token_upd()
RETURNS trigger AS $$
BEGIN
    NEW.updated := current_timestamp;
    RETURN NEW;
END
$$ LANGUAGE 'plpgsql';


CREATE TRIGGER t_ymux_auth_token_trig
    BEFORE update ON "t_ymux_auth_token"
    FOR EACH ROW
    EXECUTE PROCEDURE t_ymux_auth_token_upd();
```

Run the file `hw26_03.sql`

```
CREATE SEQUENCE t_log_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 9223372036854775807
  START 1
  CACHE 1;

DROP TABLE if exists "t_ymux_user_log" cascade ;

CREATE TABLE "t_ymux_user_log" (
      "id"                    uuid DEFAULT uuid_generate_v4() not null primary key
    , "user_id"               uuid
    , "seq"                   bigint DEFAULT nextval('t_log_seq'::regclass) NOT NULL
    , "activity_name"     text
    , "updated"           timestamp
    , "created"           timestamp default current_timestamp not null
);

create index "t_ymux_user_log_p1" on "t_ymux_user_log" ( "user_id", "seq" );
create index "t_ymux_user_log_p2" on "t_ymux_user_log" ( "user_id", "created" );

ALTER TABLE "t_ymux_user_log"
    ADD CONSTRAINT "t_ymux_user_log_user_id_fk1"
    FOREIGN KEY ("user_id")
    REFERENCES "t_ymux_user" ("id")
;

CREATE OR REPLACE function t_ymux_user_log_upd()
RETURNS trigger AS $$
```

```
BEGIN
    NEW.updated := current_timestamp;
    RETURN NEW;
END
$$ LANGUAGE 'plpgsql';


CREATE TRIGGER t_ymux_user_log_trig
    BEFORE update ON "t_ymux_user_log"
    FOR EACH ROW
    EXECUTE PROCEDURE t_ymux_user_log_upd();
```

Applications Tables

"t_ymux_user" joins to ct_login on a 1 to 1 basis. Each time a user is inserted tinto ct_login a set of rows is populated for all the homework that exits into ct_homework_ans.

ct_login

Run the file `hw26_04.sql`

```
DROP TABLE if exists ct_login cascade ;

CREATE TABLE ct_login (
      user_id                   uuid not null primary key -- 1 to 1 to "t_ymux_user"."id"
    , pg_acct                   char varying (20) not null
    , class_no                  text default '4010-BC' not null -- 4820 or 4010-BC - one of
    , lang_to_use               text default 'Go' not null      -- Go or PostgreSQL
    , misc                      JSONb default '{}' not null     -- Whatever I forgot
);

create unique index ct_login_u1 on ct_login ( pg_acct );
create index ct_login_p1 on ct_login using gin ( misc );

ALTER TABLE ct_login
    ADD CONSTRAINT ct_login_user_id_fk
    FOREIGN KEY (user_id)
    REFERENCES "t_ymux_user" ("id")
;
```

ct_homeowrk

Run the file `hw26_05.sql`

```
DROP TABLE if exists ct_homework cascade;

CREATE TABLE ct_homework (
      homework_id               uuid DEFAULT uuid_generate_v4() not null primary key
    , homework_title            text not null
    , homework_no                   text not null
    , points_avail              int not null default 10
    , video_url                 text not null
    , video_img                 text not null
    , lesson_body               JSONb not null  -- body, html, text etc.
);

CREATE INDEX ct_homework_p1 on ct_homework ( homework_no );
```

ct__homeowrk__ans

Run the file `hw26_06.sql`

```
DROP TABLE if exists ct_homework_ans cascade ;

CREATE TABLE ct_homework_ans (
      homework_ans_id           uuid DEFAULT uuid_generate_v4() not null primary key
    , homework_id               uuid not null
    , user_id                   uuid not null
    , points                    int not null default 0
    , completed                 char(1) default 'n' not null
    , updated                   timestamp
    , created                   timestamp default current_timestamp not null
);


create unique index ct_homework_ans_u1 on ct_homework_ans ( homework_id, user_id );
create unique index ct_homework_ans_u2 on ct_homework_ans ( user_id, homework_id );


-- homework_id is fk to ct_homework
ALTER TABLE ct_homework_ans
    ADD CONSTRAINT homework_id_fk
    FOREIGN KEY (homework_id)
    REFERENCES ct_homework (homework_id)
;

-- user_id is fk to ct_login
ALTER TABLE ct_homework_ans
    ADD CONSTRAINT user_id_fk
```

```
    FOREIGN KEY (user_id)
    REFERENCES ct_login (user_id)
;


CREATE OR REPLACE function ct_homework_ans_upd()
RETURNS trigger AS $$
BEGIN
    NEW.updated := current_timestamp;
    RETURN NEW;
END
$$ LANGUAGE 'plpgsql';


CREATE TRIGGER ct_homework_ans_trig
    BEFORE update ON ct_homework_ans
    FOR EACH ROW
    EXECUTE PROCEDURE ct_homework_ans_upd();
```

Triggers that depend on multiple tables

Run the file `hw26_07.sql`

```
CREATE OR REPLACE function ct_homework_ins()
RETURNS trigger AS $$
BEGIN
    insert into ct_homework_ans (
        user_id,
        homework_id
    ) select
        t1.user_id,
        NEW.homework_id
    from ct_login as t1
    where not exists (
            select 1 as "found"
            from ct_homework_ans t2
            where t2.user_id = t1.user_id
              and t2.homework_id = NEW.homework_id
        )
    ;
    RETURN NEW;
END
$$ LANGUAGE 'plpgsql';
```

```
DROP TRIGGER if exists ct_homework_trig_ins_upd on ct_homework;

CREATE TRIGGER ct_homework_trig_ins_upd
    AFTER insert or update ON ct_homework
    FOR EACH ROW
    EXECUTE PROCEDURE ct_homework_ins();
```

```
CREATE OR REPLACE function ct_homework_del()
RETURNS trigger AS $$
BEGIN
    update ct_homework_ans t3
        set completed = 'x'
        where t3.homework_id = NEW.homework_id
          and t3.completed = 'n'
    ;
    RETURN OLD;
END
$$ LANGUAGE 'plpgsql';


DROP TRIGGER if exists ct_homework_trig_del on ct_homework;

CREATE TRIGGER ct_homework_trig_del
    BEFORE delete ON ct_homework
    FOR EACH ROW
    EXECUTE PROCEDURE ct_homework_del();
```

```
CREATE OR REPLACE function ct_login_ins()
RETURNS trigger AS $$
BEGIN
    insert into ct_homework_ans (
        user_id,
        homework_id
```

```
    ) select
        NEW.user_id,
        t1.homework_id
    from ct_homework as t1
    where not exists (
            select 1 as "found"
            from ct_homework_ans t2
            where t2.user_id = NEW.user_id
              and t2.homework_id = t1.homework_id
        )
    ;
    RETURN NEW;
END
$$ LANGUAGE 'plpgsql';


drop TRIGGER if exists ct_login_trig on ct_login;

CREATE TRIGGER ct_login_trig
    AFTER insert ON ct_login
    FOR EACH ROW
    EXECUTE PROCEDURE ct_login_ins();
```

Homework tags

Run the file `hw26_09.sql` and `hw26_10.sql`

```
DROP TABLE IF EXISTS ct_tag cascade ;

CREATE TABLE ct_tag (
    tag_id uuid DEFAULT uuid_generate_v4() not null primary key,
    tag_word text not null
);

CREATE UNIQUE INDEX ct_tag_p1 on ct_tag ( tag_word );



DROP TABLE IF EXISTS ct_tag_homework cascade ;

CREATE TABLE ct_tag_homework (
    tag_id      uuid not null,
    homework_id uuid not null,
    primary key ( homework_id, tag_id )
);
```

```
CREATE UNIQUE INDEX ct_tag_homework_u1 on ct_tag_homework ( tag_id, homework_id );
```

Setup default data

Run the file `hw26_11.sql` to setup data for these tables.

```
select setup_data_26();
```