Семестровая работа по Базам Данных

Список заголовок.

- **Описание предметной области БД**
- **Описание концептуальной и логической моделей**
- **Код создания таблиц**
- **Логическое описание SQL запросов и их код с комментариями**
- **Логическое описание функции и триггера и их код с комментариями**
- **Описание ролевой модели и код создания ролевой модели**

Описание предметной области БД.

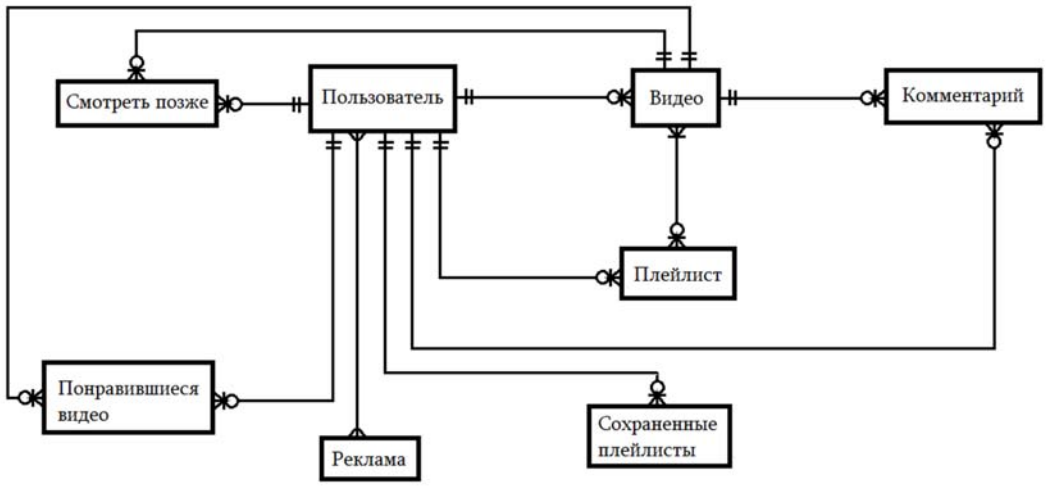
Данная база данных симулирует БД Ютуба. С ее помощью можно хранить работать с информацией о видео, пользователей, плейлистов и так далее.

Описание концептуальной и логической моделей.

В концептуальной модели приведены основные элементы БД — таблицы

- пользователь,
- видео,
- комментарий,
- плейлист,
- сохраненные плейлисты,
- смотреть позже,
- понравившиеся видео,
- реклама.

Концептуальная модель БД Ютуба



В логической модели появляется еще одна таблица для нормализации базы:

- видео-пользователь.

Код создания таблиц

```
create schema youtube;

create table youtube.user_profile (
    user_id             integer primary key,
    first_nm            varchar(255) not null,
    last_nm             varchar(255) not null,
    gender              varchar(12) not null,
    birth_date          date not null,
    hometown            varchar(255)
);

create table youtube.video (
    video_id        integer primary key,
    user_id         integer not null,
    title           varchar(255) not null,
    description     varchar(1000),
    category        varchar(255),
    like_cnt        integer default 0,
    dislike_cnt     integer default 0,
    view_cnt        integer default 0,
    comment_cnt     integer default 0,
    date_created    date default now()::date,
    duration        integer check (duration >= 3 and duration <= 10000), --in seconds
    constraint  FK_user_video
            foreign key (user_id)
            references youtube.user_profile(user_id)
);

create index video_title_index on youtube.video (title);

create table youtube.comment (
    comment_id      integer primary key,
    user_id         integer not null,
    video_id        integer not null,
    edited          bool default false,
    comment_text    varchar(1000) not null,
    date_created    date default now()::date,
    constraint FK_user_comment
            foreign key (user_id)
            references youtube.user_profile(user_id),
    constraint FK_video_comment
            foreign key (video_id)
            references youtube.video(video_id)
);
```

```sql
create table youtube.playlist (
    playlist_id     integer primary key,
    user_id         integer not null,
    is_private      bool not null,
    view_cnt        integer default 0,
    playlist_name   varchar(250),
    constraint FK_user_playlist
            foreign key (user_id)
            references youtube.user_profile(user_id)
);

create table youtube.video_playlist (
    playlist_id         integer not null,
    video_id            integer not null,
    add_count           integer default 1,
    delete_count        integer default 0,
    date_added          date default now()::date,
    date_deleted        date default null,
    constraint PK_video_playlist
            primary key (video_id, playlist_id, add_count, delete_count),
    constraint FK_playlist_video_playlist
            foreign key (playlist_id)
            references youtube.playlist(playlist_id),
    constraint FK_video_video_playlist
            foreign key (video_id)
            references youtube.video(video_id)
);

create table youtube.saved_playlists (
    user_id     integer not null,
    playlist_id integer not null,
    constraint PK_saved_playlists
            primary key (user_id, playlist_id),
    constraint FK_playlist_saved_playlists
            foreign key (playlist_id)
            references youtube.playlist(playlist_id),
    constraint FK_user_saved_playlists
            foreign key (user_id)
            references youtube.user_profile(user_id)
);
```

```sql
create table youtube.advertisement (
    user_id            integer not null,
    preferred_ad_type  varchar(255),
    constraint PK_advertisement
           primary key (user_id, preferred_ad_type),
    constraint FK_user_advertisement
           foreign key (user_id)
           references youtube.user_profile(user_id)
);

create table youtube.liked_videos (
    user_id     integer not null,
    video_id    integer not null,
    constraint PK_liked_videos
           primary key (user_id, video_id),
    constraint FK_user_liked_videos
           foreign key (user_id)
           references youtube.user_profile(user_id),
    constraint FK_video_liked_videos
           foreign key (video_id)
           references youtube.video(video_id)
);

create table youtube.watch_later (
    user_id     integer not null,
    video_id    integer not null,
    constraint PK_watch_later
           primary key (user_id, video_id),
    constraint FK_user_watch_later
           foreign key (user_id)
           references youtube.user_profile(user_id),
    constraint FK_video_watch_later
           foreign key (video_id)
           references youtube.video(video_id)
);
```

Логическое описание SQL запросов и их код с комментариями

1. Сколько пользователей предпочитают тот или иной тип рекламы?

```
--how many users prefer this or that type of ads
select count(user_id), preferred_ad_type
from youtube.advertisement
group by preferred_ad_type;
```

2. Какие пользователи имеют плейлист, в котором есть хотя бы одно видео с не менее чем 30000 просмотрами?

```
--which users have a playlist with a video with >=30000 views
with playlist_gt_1000 (playlist_id)
as (
    select distinct playlist_id
    from youtube.video_playlist
    inner join youtube.video
    on video_playlist.video_id = video.video_id
    where video.view_cnt >= 30000
    )
select distinct user_profile.user_id, first_nm, last_nm
from youtube.user_profile
inner join (
    youtube.playlist
    inner join playlist_gt_1000
    on playlist.playlist_id = playlist_gt_1000.playlist_id
    )
on user_profile.user_id = playlist.user_id;
```

3. Какие пользователи были активны в периоде от 2013 до 2015 включительно? Под активностью подразумевается то, что данный пользователь либо загрузил видео, либо написал комментарий, либо добавил видео в какой-то плейлист (в понравившиеся видео, смотреть позже или другой свой плейлист).

```
--which users have been active between 2013 and 2015 inclusive
--that is which users have
--uploaded a video OR
--commented on a video OR
--modified a playlist (
--added a video into a playlist OR
--deleted a video from a playlist
--)
select user_id, first_nm, last_nm
from
(
```

```
            select *
            from youtube.video
            where date_created < to_date('2016-01-01', 'YYYY-MM-DD')
              and to_date('2013-01-01', 'YYYY-MM-DD') <= date_created
        ) as vd
            inner join youtube.user_profile
                       on vd.user_id = user_profile.user_id
) user_upload
union
(
--users that commented on a video
    select distinct user_profile.user_id, first_nm, last_nm
    from (
            select *
            from youtube.comment
            where date_created < to_date('2016-01-01', 'YYYY-MM-DD')
              and to_date('2013-01-01', 'YYYY-MM-DD') <= date_created
        ) as cm
            inner join youtube.user_profile
                       on cm.user_id = user_profile.user_id
)
union
(
--users that modified a playlist
    select distinct user_profile.user_id, first_nm, last_nm
    from (
        (
            select distinct playlist_id
            from youtube.video_playlist
            where (date_added < to_date('2016-01-01', 'YYYY-MM-DD')
                and to_date('2013-01-01', 'YYYY-MM-DD') <= date_added)
               or (date_deleted < to_date('2016-01-01', 'YYYY-MM-DD')
                and to_date('2013-01-01', 'YYYY-MM-DD') <= date_deleted)
        ) pid
            inner join youtube.playlist
            on pid.playlist_id = playlist.playlist_id
        ) upid
    inner join youtube.user_profile
    on upid.user_id = user_profile.user_id
);
```

Логическое описание функции и триггера и их код с комментариями

Следующая функция визуализирует сколько всего просмотров имеют видео по пользователям, которые выложили их:

```
create or replace function youtube.views_per_user()
 returns table (user_id integer, count bigint) as $$
    begin
        return query (
            select youtube.video.user_id, sum(youtube.video.view_cnt)
            from youtube.video
            group by youtube.video.user_id
            );
    end
$$ language plpgsql;
drop function youtube.views_per_user();
```

При добавлении нового кортежа в таблицу video_playlist следующий триггер проверяет, данное видео удаляется, или добавляется в данный плейлист, и соответственно проставляет поле date_deleted в старых кортежах таблицы, где это поле было 9999-01-01:

```
create or replace function youtube.update_video_playlist()
returns trigger as $$
    begin
        if new.add_count = new.delete_count then
            delete from youtube.video_playlist
            where video_id = new.video_id
            and playlist_id = new.playlist_id
            and add_count = new.add_count;
        elsif new.add_count = new.delete_count + 1 then
            if new.date_deleted != to_date('9999-01-01', 'YYYY-MM-DD') then
                raise exception 'error: date deleted should be set to 9999-01-01';
            end if;
        else
            raise exception 'error: add_count - delete_count should be either 0 or 1';
        end if;
        return new;
    end;
$$ language plpgsql;


create trigger handle_video_playlist_changes
before insert on youtube.video_playlist
for each row
execute procedure youtube.update_video_playlist();
```

Описание ролевой модели и код создания ролевой модели

Имеются пользователи

- me_the_owner, который имеет все привилегии;
- info_adder, который может только добавить информацию;
- just_a_user, который лишь может использовать схему;

а также роли

- selector, который может лишь получить информацию из схемы;
- moderator, который имеет право модифицировать схему.

```
create user me_the_owner with password 'MeTheOwner';
grant all privileges on schema youtube to me_the_owner with grant option;

create user info_adder with password 'InfoAdder';
grant insert on all tables in schema youtube to info_adder with grant option;
grant create on schema youtube to info_adder with grant option;

create user just_a_user with password 'JustAUser';
grant usage on schema youtube to just_a_user;

create role selector;
grant select on all tables in schema youtube to selector;

create role moderator;
grant insert, delete, update on all tables in schema youtube to moderator;
```