```sql
--
-- PostgreSQL database dump
--

-- Dumped from database version 14.1
-- Dumped by pg_dump version 14.3

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

--
-- Name: auth; Type: SCHEMA; Schema: -; Owner: supabase_admin
--

CREATE SCHEMA auth;


ALTER SCHEMA auth OWNER TO supabase_admin;

--
-- Name: extensions; Type: SCHEMA; Schema: -; Owner: postgres
--

CREATE SCHEMA extensions;


ALTER SCHEMA extensions OWNER TO postgres;

--
-- Name: pg_graphql; Type: EXTENSION; Schema: -; Owner: -
--

CREATE EXTENSION IF NOT EXISTS pg_graphql WITH SCHEMA public;


--
-- Name: EXTENSION pg_graphql; Type: COMMENT; Schema: -; Owner:
--

COMMENT ON EXTENSION pg_graphql IS 'GraphQL support';


--
-- Name: graphql_public; Type: SCHEMA; Schema: -; Owner: supabase_admin
--

CREATE SCHEMA graphql_public;


ALTER SCHEMA graphql_public OWNER TO supabase_admin;

--
-- Name: pgbouncer; Type: SCHEMA; Schema: -; Owner: pgbouncer
--

CREATE SCHEMA pgbouncer;
```

```
65
66
67  ALTER SCHEMA pgbouncer OWNER TO pgbouncer;
68
69  --
70  -- Name: realtime; Type: SCHEMA; Schema: -; Owner: supabase_admin
71  --
72
73  CREATE SCHEMA realtime;
74
75
76  ALTER SCHEMA realtime OWNER TO supabase_admin;
77
78  --
79  -- Name: storage; Type: SCHEMA; Schema: -; Owner: supabase_admin
80  --
81
82  CREATE SCHEMA storage;
83
84
85  ALTER SCHEMA storage OWNER TO supabase_admin;
86
87  --
88  -- Name: pg_stat_statements; Type: EXTENSION; Schema: -; Owner: -
89  --
90
91  CREATE EXTENSION IF NOT EXISTS pg_stat_statements WITH SCHEMA extensions;
92
93
94  --
95  -- Name: EXTENSION pg_stat_statements; Type: COMMENT; Schema: -; Owner:
96  --
97
98  COMMENT ON EXTENSION pg_stat_statements IS 'track planning and execution statistics of
    all SQL statements executed';
99
100
101 --
102 -- Name: pgcrypto; Type: EXTENSION; Schema: -; Owner: -
103 --
104
105 CREATE EXTENSION IF NOT EXISTS pgcrypto WITH SCHEMA extensions;
106
107
108 --
109 -- Name: EXTENSION pgcrypto; Type: COMMENT; Schema: -; Owner:
110 --
111
112 COMMENT ON EXTENSION pgcrypto IS 'cryptographic functions';
113
114
115 --
116 -- Name: pgjwt; Type: EXTENSION; Schema: -; Owner: -
117 --
118
119 CREATE EXTENSION IF NOT EXISTS pgjwt WITH SCHEMA extensions;
120
121
122 --
123 -- Name: EXTENSION pgjwt; Type: COMMENT; Schema: -; Owner:
124 --
125
126 COMMENT ON EXTENSION pgjwt IS 'JSON Web Token API for Postgresql';
127
```

```
128
129  --
130  -- Name: uuid-ossp; Type: EXTENSION; Schema: -; Owner: -
131  --
132
133  CREATE EXTENSION IF NOT EXISTS "uuid-ossp" WITH SCHEMA extensions;
134
135
136  --
137  -- Name: EXTENSION "uuid-ossp"; Type: COMMENT; Schema: -; Owner:
138  --
139
140  COMMENT ON EXTENSION "uuid-ossp" IS 'generate universally unique identifiers (UUIDs)';
141
142
143  --
144  -- Name: action; Type: TYPE; Schema: realtime; Owner: supabase_admin
145  --
146
147  CREATE TYPE realtime.action AS ENUM (
148      'INSERT',
149      'UPDATE',
150      'DELETE',
151      'TRUNCATE',
152      'ERROR'
153  );
154
155
156  ALTER TYPE realtime.action OWNER TO supabase_admin;
157
158  --
159  -- Name: equality_op; Type: TYPE; Schema: realtime; Owner: supabase_admin
160  --
161
162  CREATE TYPE realtime.equality_op AS ENUM (
163      'eq',
164      'neq',
165      'lt',
166      'lte',
167      'gt',
168      'gte'
169  );
170
171
172  ALTER TYPE realtime.equality_op OWNER TO supabase_admin;
173
174  --
175  -- Name: user_defined_filter; Type: TYPE; Schema: realtime; Owner: supabase_admin
176  --
177
178  CREATE TYPE realtime.user_defined_filter AS (
179      column_name text,
180      op realtime.equality_op,
181      value text
182  );
183
184
185  ALTER TYPE realtime.user_defined_filter OWNER TO supabase_admin;
186
187  --
188  -- Name: wal_column; Type: TYPE; Schema: realtime; Owner: supabase_admin
189  --
190
191  CREATE TYPE realtime.wal_column AS (
```

```
    name text,
    type text,
    value jsonb,
    is_pkey boolean,
    is_selectable boolean
);


ALTER TYPE realtime.wal_column OWNER TO supabase_admin;

--
-- Name: wal_rls; Type: TYPE; Schema: realtime; Owner: supabase_admin
--

CREATE TYPE realtime.wal_rls AS (
    wal jsonb,
    is_rls_enabled boolean,
    subscription_ids uuid[],
    errors text[]
);


ALTER TYPE realtime.wal_rls OWNER TO supabase_admin;

--
-- Name: email(); Type: FUNCTION; Schema: auth; Owner: supabase_auth_admin
--

CREATE FUNCTION auth.email() RETURNS text
    LANGUAGE sql STABLE
    AS $$
  select
    coalesce(
        nullif(current_setting('request.jwt.claim.email', true), ''),
        (nullif(current_setting('request.jwt.claims', true), '')::jsonb ->> 'email')
    )::text
$$;


ALTER FUNCTION auth.email() OWNER TO supabase_auth_admin;

--
-- Name: role(); Type: FUNCTION; Schema: auth; Owner: supabase_auth_admin
--

CREATE FUNCTION auth.role() RETURNS text
    LANGUAGE sql STABLE
    AS $$
  select
    coalesce(
        nullif(current_setting('request.jwt.claim.role', true), ''),
        (nullif(current_setting('request.jwt.claims', true), '')::jsonb ->> 'role')
    )::text
$$;


ALTER FUNCTION auth.role() OWNER TO supabase_auth_admin;

--
-- Name: uid(); Type: FUNCTION; Schema: auth; Owner: supabase_auth_admin
--

CREATE FUNCTION auth.uid() RETURNS uuid
    LANGUAGE sql STABLE
```

```
    AS $$
  select
    coalesce(
        nullif(current_setting('request.jwt.claim.sub', true), ''),
        (nullif(current_setting('request.jwt.claims', true), '')::jsonb ->> 'sub')
    )::uuid
$$;


ALTER FUNCTION auth.uid() OWNER TO supabase_auth_admin;

--
-- Name: grant_pg_cron_access(); Type: FUNCTION; Schema: extensions; Owner: postgres
--

CREATE FUNCTION extensions.grant_pg_cron_access() RETURNS event_trigger
    LANGUAGE plpgsql
    AS $$
DECLARE
  schema_is_cron bool;
BEGIN
  schema_is_cron = (
    SELECT n.nspname = 'cron'
    FROM pg_event_trigger_ddl_commands() AS ev
    LEFT JOIN pg_catalog.pg_namespace AS n
      ON ev.objid = n.oid
  );

  IF schema_is_cron
  THEN
    grant usage on schema cron to postgres with grant option;

    alter default privileges in schema cron grant all on tables to postgres with grant option;
    alter default privileges in schema cron grant all on functions to postgres with grant option;
    alter default privileges in schema cron grant all on sequences to postgres with grant option;

    alter default privileges for user supabase_admin in schema cron grant all
        on sequences to postgres with grant option;
    alter default privileges for user supabase_admin in schema cron grant all
        on tables to postgres with grant option;
    alter default privileges for user supabase_admin in schema cron grant all
        on functions to postgres with grant option;

    grant all privileges on all tables in schema cron to postgres with grant option;

  END IF;

END;
$$;


ALTER FUNCTION extensions.grant_pg_cron_access() OWNER TO postgres;

--
-- Name: FUNCTION grant_pg_cron_access(); Type: COMMENT; Schema: extensions; Owner: postgres
--

COMMENT ON FUNCTION extensions.grant_pg_cron_access() IS 'Grants access to pg_cron';
```

```
--
-- Name: grant_pg_graphql_access(); Type: FUNCTION; Schema: extensions; Owner: supabase_admin
--

CREATE FUNCTION extensions.grant_pg_graphql_access() RETURNS event_trigger
    LANGUAGE plpgsql
    AS $_$
DECLARE
    func_is_graphql_resolve bool;
BEGIN
    func_is_graphql_resolve = (
        SELECT n.proname = 'resolve'
        FROM pg_event_trigger_ddl_commands() AS ev
        LEFT JOIN pg_catalog.pg_proc AS n
        ON ev.objid = n.oid
    );

    IF func_is_graphql_resolve
    THEN
        grant usage on schema graphql to postgres, anon, authenticated, service_role;
        grant all on function graphql.resolve to postgres, anon, authenticated, service_role;

        alter default privileges in schema graphql grant all on tables to postgres, anon, authenticated, service_role;
        alter default privileges in schema graphql grant all on functions to postgres, anon, authenticated, service_role;
        alter default privileges in schema graphql grant all on sequences to postgres, anon, authenticated, service_role;

        -- Update public wrapper to pass all arguments through to the pg_graphql resolve func
        create or replace function graphql_public.graphql(
            "operationName" text default null,
            query text default null,
            variables jsonb default null,
            extensions jsonb default null
        )
            returns jsonb
            language sql
        as $$
            -- This changed
            select graphql.resolve(
                query := query,
                variables := coalesce(variables, '{}'),
                "operationName" := "operationName",
                extensions := extensions
            );
        $$;

        grant select on graphql.field, graphql.type, graphql.enum_value to postgres, anon, authenticated, service_role;
        grant execute on function graphql.resolve to postgres, anon, authenticated, service_role;
    END IF;

END;
$_$;


ALTER FUNCTION extensions.grant_pg_graphql_access() OWNER TO supabase_admin;

--
```

```
372  -- Name: FUNCTION grant_pg_graphql_access(); Type: COMMENT; Schema: extensions; Owner:
     supabase_admin
373  --
374
375  COMMENT ON FUNCTION extensions.grant_pg_graphql_access() IS 'Grants access to
     pg_graphql';
376
377
378  --
379  -- Name: grant_pg_net_access(); Type: FUNCTION; Schema: extensions; Owner: postgres
380  --
381
382  CREATE FUNCTION extensions.grant_pg_net_access() RETURNS event_trigger
383      LANGUAGE plpgsql
384      AS $$
385  BEGIN
386    IF EXISTS (
387      SELECT 1
388      FROM pg_event_trigger_ddl_commands() AS ev
389      JOIN pg_extension AS ext
390      ON ev.objid = ext.oid
391      WHERE ext.extname = 'pg_net'
392    )
393    THEN
394      IF NOT EXISTS (
395        SELECT 1
396        FROM pg_roles
397        WHERE rolname = 'supabase_functions_admin'
398      )
399      THEN
400        CREATE USER supabase_functions_admin NOINHERIT CREATEROLE LOGIN NOREPLICATION;
401      END IF;
402
403      GRANT USAGE ON SCHEMA net TO supabase_functions_admin, postgres, anon, authenticated
       , service_role;
404
405      ALTER function net.http_get(url text, params jsonb, headers jsonb,
       timeout_milliseconds integer) SECURITY DEFINER;
406      ALTER function net.http_post(url text, body jsonb, params jsonb, headers jsonb,
       timeout_milliseconds integer) SECURITY DEFINER;
407      ALTER function net.http_collect_response(request_id bigint, async boolean) SECURITY
       DEFINER;
408
409      ALTER function net.http_get(url text, params jsonb, headers jsonb,
       timeout_milliseconds integer) SET search_path = net;
410      ALTER function net.http_post(url text, body jsonb, params jsonb, headers jsonb,
       timeout_milliseconds integer) SET search_path = net;
411      ALTER function net.http_collect_response(request_id bigint, async boolean) SET
       search_path = net;
412
413      REVOKE ALL ON FUNCTION net.http_get(url text, params jsonb, headers jsonb,
       timeout_milliseconds integer) FROM PUBLIC;
414      REVOKE ALL ON FUNCTION net.http_post(url text, body jsonb, params jsonb, headers
       jsonb, timeout_milliseconds integer) FROM PUBLIC;
415      REVOKE ALL ON FUNCTION net.http_collect_response(request_id bigint, async boolean)
       FROM PUBLIC;
416
417      GRANT EXECUTE ON FUNCTION net.http_get(url text, params jsonb, headers jsonb,
       timeout_milliseconds integer) TO supabase_functions_admin, postgres, anon,
       authenticated, service_role;
418      GRANT EXECUTE ON FUNCTION net.http_post(url text, body jsonb, params jsonb, headers
       jsonb, timeout_milliseconds integer) TO supabase_functions_admin, postgres, anon,
       authenticated, service_role;
419      GRANT EXECUTE ON FUNCTION net.http_collect_response(request_id bigint, async boolean
```

```
    ) TO supabase_functions_admin, postgres, anon, authenticated, service_role;
  END IF;
END;
$$;


ALTER FUNCTION extensions.grant_pg_net_access() OWNER TO postgres;

--
-- Name: FUNCTION grant_pg_net_access(); Type: COMMENT; Schema: extensions; Owner: postgres
--

COMMENT ON FUNCTION extensions.grant_pg_net_access() IS 'Grants access to pg_net';


--
-- Name: pgrst_ddl_watch(); Type: FUNCTION; Schema: extensions; Owner: supabase_admin
--

CREATE FUNCTION extensions.pgrst_ddl_watch() RETURNS event_trigger
    LANGUAGE plpgsql
    AS $$
DECLARE
  cmd record;
BEGIN
  FOR cmd IN SELECT * FROM pg_event_trigger_ddl_commands()
  LOOP
    IF cmd.command_tag IN (
      'CREATE SCHEMA', 'ALTER SCHEMA'
    , 'CREATE TABLE', 'CREATE TABLE AS', 'SELECT INTO', 'ALTER TABLE'
    , 'CREATE FOREIGN TABLE', 'ALTER FOREIGN TABLE'
    , 'CREATE VIEW', 'ALTER VIEW'
    , 'CREATE MATERIALIZED VIEW', 'ALTER MATERIALIZED VIEW'
    , 'CREATE FUNCTION', 'ALTER FUNCTION'
    , 'CREATE TRIGGER'
    , 'CREATE TYPE', 'ALTER TYPE'
    , 'CREATE RULE'
    , 'COMMENT'
    )
    -- don't notify in case of CREATE TEMP table or other objects created on pg_temp
    AND cmd.schema_name is distinct from 'pg_temp'
    THEN
      NOTIFY pgrst, 'reload schema';
    END IF;
  END LOOP;
END; $$;


ALTER FUNCTION extensions.pgrst_ddl_watch() OWNER TO supabase_admin;

--
-- Name: pgrst_drop_watch(); Type: FUNCTION; Schema: extensions; Owner: supabase_admin
--

CREATE FUNCTION extensions.pgrst_drop_watch() RETURNS event_trigger
    LANGUAGE plpgsql
    AS $$
DECLARE
  obj record;
BEGIN
  FOR obj IN SELECT * FROM pg_event_trigger_dropped_objects()
  LOOP
    IF obj.object_type IN (
```

```
      'schema'
    , 'table'
    , 'foreign table'
    , 'view'
    , 'materialized view'
    , 'function'
    , 'trigger'
    , 'type'
    , 'rule'
    )
    AND obj.is_temporary IS false -- no pg_temp objects
    THEN
      NOTIFY pgrst, 'reload schema';
    END IF;
  END LOOP;
END; $$;


ALTER FUNCTION extensions.pgrst_drop_watch() OWNER TO supabase_admin;

--
-- Name: set_graphql_placeholder(); Type: FUNCTION; Schema: extensions; Owner: supabase_admin
--

CREATE FUNCTION extensions.set_graphql_placeholder() RETURNS event_trigger
    LANGUAGE plpgsql
    AS $_$
    DECLARE
    graphql_is_dropped bool;
    BEGIN
    graphql_is_dropped = (
        SELECT ev.schema_name = 'graphql_public'
        FROM pg_event_trigger_dropped_objects() AS ev
        WHERE ev.schema_name = 'graphql_public'
    );

    IF graphql_is_dropped
    THEN
        create or replace function graphql_public.graphql(
            "operationName" text default null,
            query text default null,
            variables jsonb default null,
            extensions jsonb default null
        )
            returns jsonb
            language plpgsql
        as $$
            DECLARE
                server_version float;
            BEGIN
                server_version = (SELECT (SPLIT_PART((select version()), ' ', 2))::float
                );

                IF server_version >= 14 THEN
                    RETURN jsonb_build_object(
                        'errors', jsonb_build_array(
                            jsonb_build_object(
                                'message', 'pg_graphql extension is not enabled.'
                            )
                        )
                    );
                ELSE
                    RETURN jsonb_build_object(
```

```
                        'errors', jsonb_build_array(
                            jsonb_build_object(
                                'message', 'pg_graphql is only available on projects
                                running Postgres 14 onwards.'
                            )
                        )
                    );
                END IF;
            END;
        $$;
    END IF;

    END;
$_$;


ALTER FUNCTION extensions.set_graphql_placeholder() OWNER TO supabase_admin;

--
-- Name: FUNCTION set_graphql_placeholder(); Type: COMMENT; Schema: extensions; Owner:
supabase_admin
--

COMMENT ON FUNCTION extensions.set_graphql_placeholder() IS 'Reintroduces placeholder
function for graphql_public.graphql';


--
-- Name: get_auth(text); Type: FUNCTION; Schema: pgbouncer; Owner: postgres
--

CREATE FUNCTION pgbouncer.get_auth(p_usename text) RETURNS TABLE(username text, password
 text)
    LANGUAGE plpgsql SECURITY DEFINER
    AS $$
BEGIN
    RAISE WARNING 'PgBouncer auth request: %', p_usename;

    RETURN QUERY
    SELECT usename::TEXT, passwd::TEXT FROM pg_catalog.pg_shadow
    WHERE usename = p_usename;
END;
$$;


ALTER FUNCTION pgbouncer.get_auth(p_usename text) OWNER TO postgres;

--
-- Name: apply_rls(jsonb, integer); Type: FUNCTION; Schema: realtime; Owner:
supabase_admin
--

CREATE FUNCTION realtime.apply_rls(wal jsonb, max_record_bytes integer DEFAULT (1024 *
1024)) RETURNS SETOF realtime.wal_rls
    LANGUAGE plpgsql
    AS $$
      declare
        -- Regclass of the table e.g. public.notes
        entity_ regclass = (quote_ident(wal ->> 'schema') || '.' || quote_ident(wal ->>
        'table'))::regclass;

        -- I, U, D, T: insert, update ...
        action realtime.action = (
          case wal ->> 'action'
```

```
            when 'I' then 'INSERT'
            when 'U' then 'UPDATE'
            when 'D' then 'DELETE'
            else 'ERROR'
          end
        );

        -- Is row level security enabled for the table
        is_rls_enabled bool = relrowsecurity from pg_class where oid = entity_;

        subscriptions realtime.subscription[] = array_agg(subs)
          from
            realtime.subscription subs
          where
            subs.entity = entity_;

        -- Subscription vars
        roles regrole[] = array_agg(distinct us.claims_role)
          from
            unnest(subscriptions) us;

        working_role regrole;
        claimed_role regrole;
        claims jsonb;

        subscription_id uuid;
        subscription_has_access bool;
        visible_to_subscription_ids uuid[] = '{}';

        -- structured info for wal's columns
        columns realtime.wal_column[];
        -- previous identity values for update/delete
        old_columns realtime.wal_column[];

        error_record_exceeds_max_size boolean = octet_length(wal::text) >
        max_record_bytes;

        -- Primary jsonb output for record
        output jsonb;

      begin
        perform set_config('role', null, true);

        columns =
          array_agg(
            (
              x->>'name',
              x->>'type',
              realtime.cast((x->'value') #>> '{}', (x->>'type')::regtype),
              (pks ->> 'name') is not null,
              true
            )::realtime.wal_column
          )
          from
            jsonb_array_elements(wal -> 'columns') x
            left join jsonb_array_elements(wal -> 'pk') pks
              on (x ->> 'name') = (pks ->> 'name');

        old_columns =
          array_agg(
            (
              x->>'name',
              x->>'type',
              realtime.cast((x->'value') #>> '{}', (x->>'type')::regtype),
```

```
664               (pks ->> 'name') is not null,
665               true
666             )::realtime.wal_column
667           )
668           from
669             jsonb_array_elements(wal -> 'identity') x
670             left join jsonb_array_elements(wal -> 'pk') pks
671               on (x ->> 'name') = (pks ->> 'name');
672
673         for working_role in select * from unnest(roles) loop
674
675           -- Update `is_selectable` for columns and old_columns
676           columns =
677             array_agg(
678               (
679                 c.name,
680                 c.type,
681                 c.value,
682                 c.is_pkey,
683                 pg_catalog.has_column_privilege(working_role, entity_, c.name, 'SELECT')
684               )::realtime.wal_column
685             )
686             from
687               unnest(columns) c;
688
689           old_columns =
690             array_agg(
691               (
692                 c.name,
693                 c.type,
694                 c.value,
695                 c.is_pkey,
696                 pg_catalog.has_column_privilege(working_role, entity_, c.name, 'SELECT')
697               )::realtime.wal_column
698             )
699             from
700               unnest(old_columns) c;
701
702           if action <> 'DELETE' and count(1) = 0 from unnest(columns) c where c.is_pkey
              then
703             return next (
704               null,
705               is_rls_enabled,
706               -- subscriptions is already filtered by entity
707               (select array_agg(s.subscription_id) from unnest(subscriptions) as s where
                   claims_role = working_role),
708               array['Error 400: Bad Request, no primary key']
709             )::realtime.wal_rls;
710
711           -- The claims role does not have SELECT permission to the primary key of entity
712           elsif action <> 'DELETE' and sum(c.is_selectable::int) <> count(1) from unnest
              (columns) c where c.is_pkey then
713             return next (
714               null,
715               is_rls_enabled,
716               (select array_agg(s.subscription_id) from unnest(subscriptions) as s where
                   claims_role = working_role),
717               array['Error 401: Unauthorized']
718             )::realtime.wal_rls;
719
720           else
721             output = jsonb_build_object(
722               'schema', wal ->> 'schema',
723               'table', wal ->> 'table',
```

```
724                 'type', action,
725                 'commit_timestamp', to_char(
726                   (wal ->> 'timestamp')::timestamptz,
727                   'YYYY-MM-DD"T"HH24:MI:SS"Z"'
728                 ),
729                 'columns', (
730                   select
731                     jsonb_agg(
732                       jsonb_build_object(
733                         'name', pa.attname,
734                         'type', pt.typname
735                       )
736                       order by pa.attnum asc
737                     )
738                       from
739                         pg_attribute pa
740                         join pg_type pt
741                           on pa.atttypid = pt.oid
742                       where
743                         attrelid = entity_
744                         and attnum > 0
745                         and pg_catalog.has_column_privilege(working_role, entity_, pa.
                            attname, 'SELECT')
746               )
747             )
748             -- Add "record" key for insert and update
749             || case
750                 when error_record_exceeds_max_size then jsonb_build_object('record',
                    '{}'::jsonb)
751                 when action in ('INSERT', 'UPDATE') then
752                   jsonb_build_object(
753                     'record',
754                     (select jsonb_object_agg((c).name, (c).value) from unnest(columns) c
                       where (c).is_selectable)
755                   )
756                 else '{}'::jsonb
757             end
758             -- Add "old_record" key for update and delete
759             || case
760                 when error_record_exceeds_max_size then jsonb_build_object('old_record',
                    '{}'::jsonb)
761                 when action in ('UPDATE', 'DELETE') then
762                   jsonb_build_object(
763                     'old_record',
764                     (select jsonb_object_agg((c).name, (c).value) from unnest(
                      old_columns) c where (c).is_selectable)
765                   )
766                 else '{}'::jsonb
767             end;
768
769             -- Create the prepared statement
770             if is_rls_enabled and action <> 'DELETE' then
771               if (select 1 from pg_prepared_statements where name = 'walrus_rls_stmt'
                  limit 1) > 0 then
772                 deallocate walrus_rls_stmt;
773               end if;
774               execute realtime.build_prepared_statement_sql('walrus_rls_stmt', entity_,
                  columns);
775             end if;
776
777             visible_to_subscription_ids = '{}';
778
779             for subscription_id, claims in (
780                 select
```

```
781                   subs.subscription_id,
782                   subs.claims
783                 from
784                   unnest(subscriptions) subs
785                 where
786                   subs.entity = entity_
787                   and subs.claims_role = working_role
788                   and realtime.is_visible_through_filters(columns, subs.filters)
789               ) loop
790
791               if not is_rls_enabled or action = 'DELETE' then
792                 visible_to_subscription_ids = visible_to_subscription_ids ||
                  subscription_id;
793               else
794                 -- Check if RLS allows the role to see the record
795                 perform
796                   set_config('role', working_role::text, true),
797                   set_config('request.jwt.claims', claims::text, true);
798
799                 execute 'execute walrus_rls_stmt' into subscription_has_access;
800
801                 if subscription_has_access then
802                   visible_to_subscription_ids = visible_to_subscription_ids ||
                    subscription_id;
803                 end if;
804               end if;
805             end loop;
806
807             perform set_config('role', null, true);
808
809             return next (
810               output,
811               is_rls_enabled,
812               visible_to_subscription_ids,
813               case
814                 when error_record_exceeds_max_size then array['Error 413: Payload Too
                  Large']
815                 else '{}'
816               end
817             )::realtime.wal_rls;
818
819           end if;
820         end loop;
821
822         perform set_config('role', null, true);
823       end;
824       $$;
825
826
827   ALTER FUNCTION realtime.apply_rls(wal jsonb, max_record_bytes integer) OWNER TO
    supabase_admin;
828
829   --
830   -- Name: build_prepared_statement_sql(text, regclass, realtime.wal_column[]); Type:
    FUNCTION; Schema: realtime; Owner: supabase_admin
831   --
832
833   CREATE FUNCTION realtime.build_prepared_statement_sql(prepared_statement_name text,
    entity regclass, columns realtime.wal_column[]) RETURNS text
834       LANGUAGE sql
835       AS $$
836       /*
837       Builds a sql string that, if executed, creates a prepared statement to
838       tests retrive a row from *entity* by its primary key columns.
```

```
839
840       Example
841         select realtime.build_prepared_statment_sql('public.notes', '{"id"}'::text[],
            '{"bigint"}'::text[])
842       */
843         select
844       'prepare ' || prepared_statement_name || ' as
845         select
846           exists(
847             select
848               1
849             from
850               ' || entity || '
851             where
852               ' || string_agg(quote_ident(pkc.name) || '=' || quote_nullable(pkc.value #>>
                  '{}') , ' and ') || '
853           )'
854         from
855           unnest(columns) pkc
856         where
857           pkc.is_pkey
858         group by
859           entity
860       $$;
861
862
863   ALTER FUNCTION realtime.build_prepared_statement_sql(prepared_statement_name text,
    entity regclass, columns realtime.wal_column[]) OWNER TO supabase_admin;
864
865   --
866   -- Name: cast(text, regtype); Type: FUNCTION; Schema: realtime; Owner: supabase_admin
867   --
868
869   CREATE FUNCTION realtime."cast"(val text, type_ regtype) RETURNS jsonb
870       LANGUAGE plpgsql IMMUTABLE
871       AS $$
872       declare
873         res jsonb;
874       begin
875         execute format('select to_jsonb(%L::'|| type_::text || ')', val)  into res;
876         return res;
877       end
878       $$;
879
880
881   ALTER FUNCTION realtime."cast"(val text, type_ regtype) OWNER TO supabase_admin;
882
883   --
884   -- Name: check_equality_op(realtime.equality_op, regtype, text, text); Type: FUNCTION;
    Schema: realtime; Owner: supabase_admin
885   --
886
887   CREATE FUNCTION realtime.check_equality_op(op realtime.equality_op, type_ regtype, val_1
     text, val_2 text) RETURNS boolean
888       LANGUAGE plpgsql IMMUTABLE
889       AS $$
890       /*
891       Casts *val_1* and *val_2* as type *type_* and check the *op* condition for truthiness
892       */
893       declare
894         op_symbol text = (
895           case
896             when op = 'eq' then '='
897             when op = 'neq' then '!='
```

```
            when op = 'lt' then '<'
            when op = 'lte' then '<='
            when op = 'gt' then '>'
            when op = 'gte' then '>='
            else 'UNKNOWN OP'
          end
        );
        res boolean;
      begin
        execute format('select %L::'|| type_::text || ' ' || op_symbol || ' %L::'|| type_
        ::text, val_1, val_2) into res;
        return res;
      end;
      $$;


  ALTER FUNCTION realtime.check_equality_op(op realtime.equality_op, type_ regtype, val_1
  text, val_2 text) OWNER TO supabase_admin;

  --
  -- Name: is_visible_through_filters(realtime.wal_column[],
  realtime.user_defined_filter[]); Type: FUNCTION; Schema: realtime; Owner: supabase_admin
  --

  CREATE FUNCTION realtime.is_visible_through_filters(columns realtime.wal_column[],
  filters realtime.user_defined_filter[]) RETURNS boolean
      LANGUAGE sql IMMUTABLE
      AS $$
      /*
      Should the record be visible (true) or filtered out (false) after *filters* are
      applied
      */
      select
        -- Default to allowed when no filters present
        coalesce(
          sum(
            realtime.check_equality_op(
              op:=f.op,
              type_:=col.type::regtype,
              -- cast jsonb to text
              val_1:=col.value #>> '{}',
              val_2:=f.value
            )::int
          ) = count(1),
          true
        )
      from
        unnest(filters) f
        join unnest(columns) col
            on f.column_name = col.name;
      $$;


  ALTER FUNCTION realtime.is_visible_through_filters(columns realtime.wal_column[],
  filters realtime.user_defined_filter[]) OWNER TO supabase_admin;

  --
  -- Name: quote_wal2json(regclass); Type: FUNCTION; Schema: realtime; Owner:
  supabase_admin
  --

  CREATE FUNCTION realtime.quote_wal2json(entity regclass) RETURNS text
      LANGUAGE sql IMMUTABLE STRICT
      AS $$
```

```
955         select
956           (
957             select string_agg('' || ch,'')
958             from unnest(string_to_array(nsp.nspname::text, null)) with ordinality x(ch,
                idx)
959             where
960               not (x.idx = 1 and x.ch = '"')
961               and not (
962                 x.idx = array_length(string_to_array(nsp.nspname::text, null), 1)
963                 and x.ch = '"'
964               )
965           )
966           || '.'
967           || (
968             select string_agg('' || ch,'')
969             from unnest(string_to_array(pc.relname::text, null)) with ordinality x(ch, idx)
970             where
971               not (x.idx = 1 and x.ch = '"')
972               and not (
973                 x.idx = array_length(string_to_array(nsp.nspname::text, null), 1)
974                 and x.ch = '"'
975               )
976           )
977         from
978           pg_class pc
979           join pg_namespace nsp
980             on pc.relnamespace = nsp.oid
981         where
982           pc.oid = entity
983       $$;
984
985
986   ALTER FUNCTION realtime.quote_wal2json(entity regclass) OWNER TO supabase_admin;
987
988   --
989   -- Name: subscription_check_filters(); Type: FUNCTION; Schema: realtime; Owner:
      supabase_admin
990   --
991
992   CREATE FUNCTION realtime.subscription_check_filters() RETURNS trigger
993       LANGUAGE plpgsql
994       AS $$
995       /*
996       Validates that the user defined filters for a subscription:
997       - refer to valid columns that the claimed role may access
998       - values are coercable to the correct column type
999       */
1000      declare
1001        col_names text[] = coalesce(
1002          array_agg(c.column_name order by c.ordinal_position),
1003          '{}'::text[]
1004        )
1005        from
1006          information_schema.columns c
1007        where
1008          format('%I.%I', c.table_schema, c.table_name)::regclass = new.entity
1009          and pg_catalog.has_column_privilege(
1010            (new.claims ->> 'role'),
1011            format('%I.%I', c.table_schema, c.table_name)::regclass,
1012            c.column_name,
1013            'SELECT'
1014          );
1015        filter realtime.user_defined_filter;
1016        col_type regtype;
```

```
      begin
        for filter in select * from unnest(new.filters) loop
          -- Filtered column is valid
          if not filter.column_name = any(col_names) then
            raise exception 'invalid column for filter %', filter.column_name;
          end if;

          -- Type is sanitized and safe for string interpolation
          col_type = (
            select atttypid::regtype
            from pg_catalog.pg_attribute
            where attrelid = new.entity
              and attname = filter.column_name
          );
          if col_type is null then
            raise exception 'failed to lookup type for column %', filter.column_name;
          end if;
          -- raises an exception if value is not coercable to type
          perform realtime.cast(filter.value, col_type);
        end loop;

        -- Apply consistent order to filters so the unique constraint on
        -- (subscription_id, entity, filters) can't be tricked by a different filter order
        new.filters = coalesce(
          array_agg(f order by f.column_name, f.op, f.value),
          '{}'
        ) from unnest(new.filters) f;

      return new;
    end;
    $$;


  ALTER FUNCTION realtime.subscription_check_filters() OWNER TO supabase_admin;

  --
  -- Name: to_regrole(text); Type: FUNCTION; Schema: realtime; Owner: supabase_admin
  --

  CREATE FUNCTION realtime.to_regrole(role_name text) RETURNS regrole
      LANGUAGE sql IMMUTABLE
      AS $$ select role_name::regrole $$;


  ALTER FUNCTION realtime.to_regrole(role_name text) OWNER TO supabase_admin;

  --
  -- Name: extension(text); Type: FUNCTION; Schema: storage; Owner: supabase_storage_admin
  --

  CREATE FUNCTION storage.extension(name text) RETURNS text
      LANGUAGE plpgsql
      AS $$
  DECLARE
  _parts text[];
  _filename text;
  BEGIN
      select string_to_array(name, '/') into _parts;
      select _parts[array_length(_parts,1)] into _filename;
      -- @todo return the last part instead of 2
      return split_part(_filename, '.', 2);
  END
  $$;
```

```
1081
1082  ALTER FUNCTION storage.extension(name text) OWNER TO supabase_storage_admin;
1083
1084  --
1085  -- Name: filename(text); Type: FUNCTION; Schema: storage; Owner: supabase_storage_admin
1086  --
1087
1088  CREATE FUNCTION storage.filename(name text) RETURNS text
1089      LANGUAGE plpgsql
1090      AS $$
1091  DECLARE
1092  _parts text[];
1093  BEGIN
1094      select string_to_array(name, '/') into _parts;
1095      return _parts[array_length(_parts,1)];
1096  END
1097  $$;
1098
1099
1100  ALTER FUNCTION storage.filename(name text) OWNER TO supabase_storage_admin;
1101
1102  --
1103  -- Name: foldername(text); Type: FUNCTION; Schema: storage; Owner: supabase_storage_admin
1104  --
1105
1106  CREATE FUNCTION storage.foldername(name text) RETURNS text[]
1107      LANGUAGE plpgsql
1108      AS $$
1109  DECLARE
1110  _parts text[];
1111  BEGIN
1112      select string_to_array(name, '/') into _parts;
1113      return _parts[1:array_length(_parts,1)-1];
1114  END
1115  $$;
1116
1117
1118  ALTER FUNCTION storage.foldername(name text) OWNER TO supabase_storage_admin;
1119
1120  --
1121  -- Name: get_size_by_bucket(); Type: FUNCTION; Schema: storage; Owner:
     supabase_storage_admin
1122  --
1123
1124  CREATE FUNCTION storage.get_size_by_bucket() RETURNS TABLE(size bigint, bucket_id text)
1125      LANGUAGE plpgsql
1126      AS $$
1127  BEGIN
1128      return query
1129          select sum((metadata->>'size')::int) as size, obj.bucket_id
1130          from "storage".objects as obj
1131          group by obj.bucket_id;
1132  END
1133  $$;
1134
1135
1136  ALTER FUNCTION storage.get_size_by_bucket() OWNER TO supabase_storage_admin;
1137
1138  --
1139  -- Name: search(text, text, integer, integer, integer, text, text, text); Type:
     FUNCTION; Schema: storage; Owner: supabase_storage_admin
1140  --
1141
1142  CREATE FUNCTION storage.search(prefix text, bucketname text, limits integer DEFAULT 100,
```

```
 levels integer DEFAULT 1, offsets integer DEFAULT 0, search text DEFAULT ''::text,
sortcolumn text DEFAULT 'name'::text, sortorder text DEFAULT 'asc'::text) RETURNS TABLE(
name text, id uuid, updated_at timestamp with time zone, created_at timestamp with time
zone, last_accessed_at timestamp with time zone, metadata jsonb)
    LANGUAGE plpgsql STABLE
    AS $_$
declare
  v_order_by text;
  v_sort_order text;
begin
  case
    when sortcolumn = 'name' then
      v_order_by = 'name';
    when sortcolumn = 'updated_at' then
      v_order_by = 'updated_at';
    when sortcolumn = 'created_at' then
      v_order_by = 'created_at';
    when sortcolumn = 'last_accessed_at' then
      v_order_by = 'last_accessed_at';
    else
      v_order_by = 'name';
  end case;

  case
    when sortorder = 'asc' then
      v_sort_order = 'asc';
    when sortorder = 'desc' then
      v_sort_order = 'desc';
    else
      v_sort_order = 'asc';
  end case;

  v_order_by = v_order_by || ' ' || v_sort_order;

  return query execute
    'with folders as (
       select path_tokens[$1] as folder
       from storage.objects
         where objects.name ilike $2 || $3 || ''%''
           and bucket_id = $4
           and array_length(regexp_split_to_array(objects.name, ''/''), 1) <> $1
       group by folder
       order by folder ' || v_sort_order || '
     )
     (select folder as "name",
            null as id,
            null as updated_at,
            null as created_at,
            null as last_accessed_at,
            null as metadata from folders)
     union all
     (select path_tokens[$1] as "name",
            id,
            updated_at,
            created_at,
            last_accessed_at,
            metadata
     from storage.objects
     where objects.name ilike $2 || $3 || ''%''
       and bucket_id = $4
       and array_length(regexp_split_to_array(objects.name, ''/''), 1) = $1
     order by ' || v_order_by || ')
     limit $5
     offset $6' using levels, prefix, search, bucketname, limits, offsets;
```

```
end;
$_$;


ALTER FUNCTION storage.search(prefix text, bucketname text, limits integer, levels
integer, offsets integer, search text, sortcolumn text, sortorder text) OWNER TO
supabase_storage_admin;

SET default_tablespace = '';

SET default_table_access_method = heap;

--
-- Name: audit_log_entries; Type: TABLE; Schema: auth; Owner: supabase_auth_admin
--

CREATE TABLE auth.audit_log_entries (
    instance_id uuid,
    id uuid NOT NULL,
    payload json,
    created_at timestamp with time zone
);


ALTER TABLE auth.audit_log_entries OWNER TO supabase_auth_admin;

--
-- Name: TABLE audit_log_entries; Type: COMMENT; Schema: auth; Owner: supabase_auth_admin
--

COMMENT ON TABLE auth.audit_log_entries IS 'Auth: Audit trail for user actions.';


--
-- Name: identities; Type: TABLE; Schema: auth; Owner: supabase_auth_admin
--

CREATE TABLE auth.identities (
    id text NOT NULL,
    user_id uuid NOT NULL,
    identity_data jsonb NOT NULL,
    provider text NOT NULL,
    last_sign_in_at timestamp with time zone,
    created_at timestamp with time zone,
    updated_at timestamp with time zone
);


ALTER TABLE auth.identities OWNER TO supabase_auth_admin;

--
-- Name: TABLE identities; Type: COMMENT; Schema: auth; Owner: supabase_auth_admin
--

COMMENT ON TABLE auth.identities IS 'Auth: Stores identities associated to a user.';


--
-- Name: instances; Type: TABLE; Schema: auth; Owner: supabase_auth_admin
--

CREATE TABLE auth.instances (
    id uuid NOT NULL,
    uuid uuid,
```

```
    raw_base_config text,
    created_at timestamp with time zone,
    updated_at timestamp with time zone
);


ALTER TABLE auth.instances OWNER TO supabase_auth_admin;

--
-- Name: TABLE instances; Type: COMMENT; Schema: auth; Owner: supabase_auth_admin
--

COMMENT ON TABLE auth.instances IS 'Auth: Manages users across multiple sites.';


--
-- Name: refresh_tokens; Type: TABLE; Schema: auth; Owner: supabase_auth_admin
--

CREATE TABLE auth.refresh_tokens (
    instance_id uuid,
    id bigint NOT NULL,
    token character varying(255),
    user_id character varying(255),
    revoked boolean,
    created_at timestamp with time zone,
    updated_at timestamp with time zone,
    parent character varying(255)
);


ALTER TABLE auth.refresh_tokens OWNER TO supabase_auth_admin;

--
-- Name: TABLE refresh_tokens; Type: COMMENT; Schema: auth; Owner: supabase_auth_admin
--

COMMENT ON TABLE auth.refresh_tokens IS 'Auth: Store of tokens used to refresh JWT
tokens once they expire.';


--
-- Name: refresh_tokens_id_seq; Type: SEQUENCE; Schema: auth; Owner: supabase_auth_admin
--

CREATE SEQUENCE auth.refresh_tokens_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE auth.refresh_tokens_id_seq OWNER TO supabase_auth_admin;

--
-- Name: refresh_tokens_id_seq; Type: SEQUENCE OWNED BY; Schema: auth; Owner:
supabase_auth_admin
--

ALTER SEQUENCE auth.refresh_tokens_id_seq OWNED BY auth.refresh_tokens.id;


--
```

```
-- Name: schema_migrations; Type: TABLE; Schema: auth; Owner: supabase_auth_admin
--

CREATE TABLE auth.schema_migrations (
    version character varying(255) NOT NULL
);


ALTER TABLE auth.schema_migrations OWNER TO supabase_auth_admin;

--
-- Name: TABLE schema_migrations; Type: COMMENT; Schema: auth; Owner: supabase_auth_admin
--

COMMENT ON TABLE auth.schema_migrations IS 'Auth: Manages updates to the auth system.';


--
-- Name: users; Type: TABLE; Schema: auth; Owner: supabase_auth_admin
--

CREATE TABLE auth.users (
    instance_id uuid,
    id uuid NOT NULL,
    aud character varying(255),
    role character varying(255),
    email character varying(255),
    encrypted_password character varying(255),
    email_confirmed_at timestamp with time zone,
    invited_at timestamp with time zone,
    confirmation_token character varying(255),
    confirmation_sent_at timestamp with time zone,
    recovery_token character varying(255),
    recovery_sent_at timestamp with time zone,
    email_change_token_new character varying(255),
    email_change character varying(255),
    email_change_sent_at timestamp with time zone,
    last_sign_in_at timestamp with time zone,
    raw_app_meta_data jsonb,
    raw_user_meta_data jsonb,
    is_super_admin boolean,
    created_at timestamp with time zone,
    updated_at timestamp with time zone,
    phone character varying(15) DEFAULT NULL::character varying,
    phone_confirmed_at timestamp with time zone,
    phone_change character varying(15) DEFAULT ''::character varying,
    phone_change_token character varying(255) DEFAULT ''::character varying,
    phone_change_sent_at timestamp with time zone,
    confirmed_at timestamp with time zone GENERATED ALWAYS AS (LEAST(email_confirmed_at,
     phone_confirmed_at)) STORED,
    email_change_token_current character varying(255) DEFAULT ''::character varying,
    email_change_confirm_status smallint DEFAULT 0,
    banned_until timestamp with time zone,
    reauthentication_token character varying(255) DEFAULT ''::character varying,
    reauthentication_sent_at timestamp with time zone,
    CONSTRAINT users_email_change_confirm_status_check CHECK (((
    email_change_confirm_status >= 0) AND (email_change_confirm_status <= 2)))
);


ALTER TABLE auth.users OWNER TO supabase_auth_admin;

--
-- Name: TABLE users; Type: COMMENT; Schema: auth; Owner: supabase_auth_admin
```

```
--

COMMENT ON TABLE auth.users IS 'Auth: Stores user login data within a secure schema.';


--
-- Name: Decisions; Type: TABLE; Schema: public; Owner: supabase_admin
--

CREATE TABLE public."Decisions" (
    id bigint NOT NULL,
    project_id bigint,
    voting_result double precision,
    is_project_accepted boolean,
    date_time_timezone timestamp with time zone
);


ALTER TABLE public."Decisions" OWNER TO supabase_admin;

--
-- Name: COLUMN "Decisions".voting_result; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Decisions".voting_result IS 'Percentage (out of 100) of accept votes among all the SingleVotes for that project (ex: 80)';


--
-- Name: COLUMN "Decisions".is_project_accepted; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Decisions".is_project_accepted IS 'Accept or Reject decision. if true, if means the project proposal is accepted.';


--
-- Name: COLUMN "Decisions".date_time_timezone; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Decisions".date_time_timezone IS 'Date, time, and timezone of when the voting decision has been reached.';


--
-- Name: Decision_id_seq; Type: SEQUENCE; Schema: public; Owner: supabase_admin
--

ALTER TABLE public."Decisions" ALTER COLUMN id ADD GENERATED BY DEFAULT AS IDENTITY (
    SEQUENCE NAME public."Decision_id_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);


--
-- Name: Donations; Type: TABLE; Schema: public; Owner: supabase_admin
--
```

```
1447
1448  CREATE TABLE public."Donations" (
1449      id bigint NOT NULL,
1450      donor_id bigint,
1451      project_id bigint,
1452      proof_of_completion_file character varying,
1453      project_completion_date_time_timezone timestamp with time zone
1454  );
1455
1456
1457  ALTER TABLE public."Donations" OWNER TO supabase_admin;
1458
1459  --
1460  -- Name: COLUMN "Donations".proof_of_completion_file; Type: COMMENT; Schema: public;
     Owner: supabase_admin
1461  --
1462
1463  COMMENT ON COLUMN public."Donations".proof_of_completion_file IS 'The name of the file
     which shows proof that the project has been completed.';
1464
1465
1466  --
1467  -- Name: COLUMN "Donations".project_completion_date_time_timezone; Type: COMMENT;
     Schema: public; Owner: supabase_admin
1468  --
1469
1470  COMMENT ON COLUMN public."Donations".project_completion_date_time_timezone IS 'Date,
     time, and timezone of when the project was completed, as taken from the time of
     submission of the proof of completion document.';
1471
1472
1473  --
1474  -- Name: Donation_id_seq; Type: SEQUENCE; Schema: public; Owner: supabase_admin
1475  --
1476
1477  ALTER TABLE public."Donations" ALTER COLUMN id ADD GENERATED BY DEFAULT AS IDENTITY (
1478      SEQUENCE NAME public."Donation_id_seq"
1479      START WITH 1
1480      INCREMENT BY 1
1481      NO MINVALUE
1482      NO MAXVALUE
1483      CACHE 1
1484  );
1485
1486
1487  --
1488  -- Name: Donors; Type: TABLE; Schema: public; Owner: supabase_admin
1489  --
1490
1491  CREATE TABLE public."Donors" (
1492      id bigint NOT NULL,
1493      wallet_address character varying
1494  );
1495
1496
1497  ALTER TABLE public."Donors" OWNER TO supabase_admin;
1498
1499  --
1500  -- Name: Donor_id_seq; Type: SEQUENCE; Schema: public; Owner: supabase_admin
1501  --
1502
1503  ALTER TABLE public."Donors" ALTER COLUMN id ADD GENERATED BY DEFAULT AS IDENTITY (
1504      SEQUENCE NAME public."Donor_id_seq"
1505      START WITH 1
```

```
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);


--
-- Name: Governors; Type: TABLE; Schema: public; Owner: supabase_admin
--

CREATE TABLE public."Governors" (
    id bigint NOT NULL,
    wallet_address character varying
);


ALTER TABLE public."Governors" OWNER TO supabase_admin;

--
-- Name: Governors_id_seq; Type: SEQUENCE; Schema: public; Owner: supabase_admin
--

ALTER TABLE public."Governors" ALTER COLUMN id ADD GENERATED BY DEFAULT AS IDENTITY (
    SEQUENCE NAME public."Governors_id_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);


--
-- Name: Organizations; Type: TABLE; Schema: public; Owner: supabase_admin
--

CREATE TABLE public."Organizations" (
    id bigint NOT NULL,
    name character varying,
    website character varying,
    country character varying,
    active boolean,
    other_info text,
    image_file character varying,
    wallet_address character varying
);


ALTER TABLE public."Organizations" OWNER TO supabase_admin;

--
-- Name: TABLE "Organizations"; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON TABLE public."Organizations" IS 'Charity Organizations';


--
-- Name: COLUMN "Organizations".other_info; Type: COMMENT; Schema: public; Owner:
supabase_admin
--

COMMENT ON COLUMN public."Organizations".other_info IS 'Name of the file which contains
```

```
more information about the Organization';


--
-- Name: COLUMN "Organizations".image_file; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Organizations".image_file IS 'Name of the image file for the organization. 1080x1080 resolution/size';


--
-- Name: COLUMN "Organizations".wallet_address; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Organizations".wallet_address IS 'Wallet address for the organization, to which the funding for the project will be deposited.';


--
-- Name: Organizations_id_seq; Type: SEQUENCE; Schema: public; Owner: supabase_admin
--

ALTER TABLE public."Organizations" ALTER COLUMN id ADD GENERATED BY DEFAULT AS IDENTITY (
    SEQUENCE NAME public."Organizations_id_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);


--
-- Name: Projects; Type: TABLE; Schema: public; Owner: supabase_admin
--

CREATE TABLE public."Projects" (
    id bigint NOT NULL,
    solution_id bigint,
    organization_id bigint,
    budget_usd double precision,
    other_info text,
    country character varying,
    project_duration_days bigint,
    status character varying,
    "date_time_timezone " timestamp with time zone,
    "mintPriceHBAR" double precision,
    "maxNFTSupply" bigint
);


ALTER TABLE public."Projects" OWNER TO supabase_admin;

--
-- Name: TABLE "Projects"; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON TABLE public."Projects" IS 'Projects created by organizations implementing solutions';


```

```
--
-- Name: COLUMN "Projects".country; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Projects".country IS 'Country where the project is to be applied';


--
-- Name: COLUMN "Projects".project_duration_days; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Projects".project_duration_days IS 'Number of days that it will take for the project to be completed';


--
-- Name: COLUMN "Projects".status; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Projects".status IS 'Possible Values: UnderReview, Rejected, Accepted, Initiated, Uninitiated, Completed, Incomplete';


--
-- Name: COLUMN "Projects"."date_time_timezone "; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Projects"."date_time_timezone " IS 'Date, time and timezone of when the project was submitted for voting';


--
-- Name: COLUMN "Projects"."mintPriceHBAR"; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Projects"."mintPriceHBAR" IS 'donation amount in HBAR to mint a single donation NFT';


--
-- Name: COLUMN "Projects"."maxNFTSupply"; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Projects"."maxNFTSupply" IS 'max number of donation NFTs for completion the full donation for the solution';


--
-- Name: Projects_id_seq; Type: SEQUENCE; Schema: public; Owner: supabase_admin
--

ALTER TABLE public."Projects" ALTER COLUMN id ADD GENERATED BY DEFAULT AS IDENTITY (
    SEQUENCE NAME public."Projects_id_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);
```

```
1681
1682
1683    --
1684    -- Name: SingleVotes; Type: TABLE; Schema: public; Owner: supabase_admin
1685    --
1686
1687    CREATE TABLE public."SingleVotes" (
1688        id bigint NOT NULL,
1689        governor_id bigint,
1690        vote_value bigint,
1691        project_id bigint,
1692        date_time_timezone timestamp with time zone
1693    );
1694
1695
1696    ALTER TABLE public."SingleVotes" OWNER TO supabase_admin;
1697
1698    --
1699    -- Name: COLUMN "SingleVotes".governor_id; Type: COMMENT; Schema: public; Owner:
        supabase_admin
1700    --
1701
1702    COMMENT ON COLUMN public."SingleVotes".governor_id IS 'id of the Governor who is
        casting the vote';
1703
1704
1705    --
1706    -- Name: COLUMN "SingleVotes".vote_value; Type: COMMENT; Schema: public; Owner:
        supabase_admin
1707    --
1708
1709    COMMENT ON COLUMN public."SingleVotes".vote_value IS 'Value of the vote. Initially
        either 0 or 1, where 0 means reject and 1 means accept';
1710
1711
1712    --
1713    -- Name: COLUMN "SingleVotes".project_id; Type: COMMENT; Schema: public; Owner:
        supabase_admin
1714    --
1715
1716    COMMENT ON COLUMN public."SingleVotes".project_id IS 'Project which is under vote';
1717
1718
1719    --
1720    -- Name: COLUMN "SingleVotes".date_time_timezone; Type: COMMENT; Schema: public; Owner:
        supabase_admin
1721    --
1722
1723    COMMENT ON COLUMN public."SingleVotes".date_time_timezone IS 'Date, time, and timezone
        of when the vote was casted';
1724
1725
1726    --
1727    -- Name: SingleVote_id_seq; Type: SEQUENCE; Schema: public; Owner: supabase_admin
1728    --
1729
1730    ALTER TABLE public."SingleVotes" ALTER COLUMN id ADD GENERATED BY DEFAULT AS IDENTITY (
1731        SEQUENCE NAME public."SingleVote_id_seq"
1732        START WITH 1
1733        INCREMENT BY 1
1734        NO MINVALUE
1735        NO MAXVALUE
1736        CACHE 1
1737    );
```

```
--
-- Name: Solutions; Type: TABLE; Schema: public; Owner: supabase_admin
--

CREATE TABLE public."Solutions" (
    id bigint NOT NULL,
    name character varying,
    category character varying,
    nature character varying,
    summary character varying,
    estimated_cost_usd double precision,
    cost_breakdown character varying,
    information_source character varying,
    application_scenarios character varying,
    applicable_in_city boolean,
    applicable_in_nature boolean,
    low_tech bigint,
    simplicity bigint,
    low_cost bigint,
    portability bigint,
    versatility bigint,
    local_materials_required character varying,
    labour_hours_required bigint,
    product_company_name character varying,
    product_webpage character varying,
    other_info text,
    applied_by_organizations character varying,
    image_file_name character varying
);


ALTER TABLE public."Solutions" OWNER TO supabase_admin;

--
-- Name: TABLE "Solutions"; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON TABLE public."Solutions" IS 'Possible solutions that can be used to help people';


--
-- Name: COLUMN "Solutions".image_file_name; Type: COMMENT; Schema: public; Owner: supabase_admin
--

COMMENT ON COLUMN public."Solutions".image_file_name IS 'Name of the image file for the Solution';


--
-- Name: Solutions_id_seq; Type: SEQUENCE; Schema: public; Owner: supabase_admin
--

ALTER TABLE public."Solutions" ALTER COLUMN id ADD GENERATED BY DEFAULT AS IDENTITY (
    SEQUENCE NAME public."Solutions_id_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);
```

```
1799
1800
1801    --
1802    -- Name: SystemConstants; Type: TABLE; Schema: public; Owner: supabase_admin
1803    --
1804
1805    CREATE TABLE public."SystemConstants" (
1806        id bigint NOT NULL,
1807        min_perc_vote double precision,
1808        path_reward_per_vote bigint,
1809        hours_to_vote bigint,
1810        days_to_claim_donation bigint,
1811        days_to_complete_project bigint
1812    );
1813
1814
1815    ALTER TABLE public."SystemConstants" OWNER TO supabase_admin;
1816
1817    --
1818    -- Name: TABLE "SystemConstants"; Type: COMMENT; Schema: public; Owner: supabase_admin
1819    --
1820
1821    COMMENT ON TABLE public."SystemConstants" IS 'Constants of the system';
1822
1823
1824    --
1825    -- Name: COLUMN "SystemConstants".min_perc_vote; Type: COMMENT; Schema: public; Owner:
        supabase_admin
1826    --
1827
1828    COMMENT ON COLUMN public."SystemConstants".min_perc_vote IS 'between 50 and 100, this
        is the percentage of votes required by a proposal to pass';
1829
1830
1831    --
1832    -- Name: COLUMN "SystemConstants".path_reward_per_vote; Type: COMMENT; Schema: public;
        Owner: supabase_admin
1833    --
1834
1835    COMMENT ON COLUMN public."SystemConstants".path_reward_per_vote IS 'number of PATH
        tokens distributed to each governor each time they submit a vote';
1836
1837
1838    --
1839    -- Name: COLUMN "SystemConstants".hours_to_vote; Type: COMMENT; Schema: public; Owner:
        supabase_admin
1840    --
1841
1842    COMMENT ON COLUMN public."SystemConstants".hours_to_vote IS 'number of hours given to
        Governors to submit a vote for a proposal, once they are invited to vote';
1843
1844
1845    --
1846    -- Name: COLUMN "SystemConstants".days_to_claim_donation; Type: COMMENT; Schema:
        public; Owner: supabase_admin
1847    --
1848
1849    COMMENT ON COLUMN public."SystemConstants".days_to_claim_donation IS 'number of days
        given to an Organization to claim the donation and start the project';
1850
1851
1852    --
1853    -- Name: COLUMN "SystemConstants".days_to_complete_project; Type: COMMENT; Schema:
        public; Owner: supabase_admin
```

```
--

COMMENT ON COLUMN public."SystemConstants".days_to_complete_project IS 'number of days given to an Organization to complete the project and submit the proof of completion report for a project, after they claimed the donation';


--
-- Name: SystemConstants_id_seq; Type: SEQUENCE; Schema: public; Owner: supabase_admin
--

ALTER TABLE public."SystemConstants" ALTER COLUMN id ADD GENERATED BY DEFAULT AS IDENTITY (
    SEQUENCE NAME public."SystemConstants_id_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);


--
-- Name: schema_migrations; Type: TABLE; Schema: realtime; Owner: supabase_admin
--

CREATE TABLE realtime.schema_migrations (
    version bigint NOT NULL,
    inserted_at timestamp(0) without time zone
);


ALTER TABLE realtime.schema_migrations OWNER TO supabase_admin;

--
-- Name: subscription; Type: TABLE; Schema: realtime; Owner: supabase_admin
--

CREATE TABLE realtime.subscription (
    id bigint NOT NULL,
    subscription_id uuid NOT NULL,
    entity regclass NOT NULL,
    filters realtime.user_defined_filter[] DEFAULT '{}'::realtime.user_defined_filter[] NOT NULL,
    claims jsonb NOT NULL,
    claims_role regrole GENERATED ALWAYS AS (realtime.to_regrole((claims ->> 'role'::text))) STORED NOT NULL,
    created_at timestamp without time zone DEFAULT timezone('utc'::text, now()) NOT NULL
);


ALTER TABLE realtime.subscription OWNER TO supabase_admin;

--
-- Name: subscription_id_seq; Type: SEQUENCE; Schema: realtime; Owner: supabase_admin
--

ALTER TABLE realtime.subscription ALTER COLUMN id ADD GENERATED ALWAYS AS IDENTITY (
    SEQUENCE NAME realtime.subscription_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
```

```
);


--
-- Name: buckets; Type: TABLE; Schema: storage; Owner: supabase_storage_admin
--

CREATE TABLE storage.buckets (
    id text NOT NULL,
    name text NOT NULL,
    owner uuid,
    created_at timestamp with time zone DEFAULT now(),
    updated_at timestamp with time zone DEFAULT now(),
    public boolean DEFAULT false
);


ALTER TABLE storage.buckets OWNER TO supabase_storage_admin;

--
-- Name: migrations; Type: TABLE; Schema: storage; Owner: supabase_storage_admin
--

CREATE TABLE storage.migrations (
    id integer NOT NULL,
    name character varying(100) NOT NULL,
    hash character varying(40) NOT NULL,
    executed_at timestamp without time zone DEFAULT CURRENT_TIMESTAMP
);


ALTER TABLE storage.migrations OWNER TO supabase_storage_admin;

--
-- Name: objects; Type: TABLE; Schema: storage; Owner: supabase_storage_admin
--

CREATE TABLE storage.objects (
    id uuid DEFAULT extensions.uuid_generate_v4() NOT NULL,
    bucket_id text,
    name text,
    owner uuid,
    created_at timestamp with time zone DEFAULT now(),
    updated_at timestamp with time zone DEFAULT now(),
    last_accessed_at timestamp with time zone DEFAULT now(),
    metadata jsonb,
    path_tokens text[] GENERATED ALWAYS AS (string_to_array(name, '/'::text)) STORED
);


ALTER TABLE storage.objects OWNER TO supabase_storage_admin;

--
-- Name: refresh_tokens id; Type: DEFAULT; Schema: auth; Owner: supabase_auth_admin
--

ALTER TABLE ONLY auth.refresh_tokens ALTER COLUMN id SET DEFAULT nextval(
'auth.refresh_tokens_id_seq'::regclass);


--
-- Data for Name: audit_log_entries; Type: TABLE DATA; Schema: auth; Owner:
supabase_auth_admin
--
```

```
1975
1976  COPY auth.audit_log_entries (instance_id, id, payload, created_at) FROM stdin;
1977  \.
1978
1979
1980  --
1981  -- Data for Name: identities; Type: TABLE DATA; Schema: auth; Owner: supabase_auth_admin
1982  --
1983
1984  COPY auth.identities (id, user_id, identity_data, provider, last_sign_in_at, created_at,
       updated_at) FROM stdin;
1985  \.
1986
1987
1988  --
1989  -- Data for Name: instances; Type: TABLE DATA; Schema: auth; Owner: supabase_auth_admin
1990  --
1991
1992  COPY auth.instances (id, uuid, raw_base_config, created_at, updated_at) FROM stdin;
1993  \.
1994
1995
1996  --
1997  -- Data for Name: refresh_tokens; Type: TABLE DATA; Schema: auth; Owner:
      supabase_auth_admin
1998  --
1999
2000  COPY auth.refresh_tokens (instance_id, id, token, user_id, revoked, created_at,
      updated_at, parent) FROM stdin;
2001  \.
2002
2003
2004  --
2005  -- Data for Name: schema_migrations; Type: TABLE DATA; Schema: auth; Owner:
      supabase_auth_admin
2006  --
2007
2008  COPY auth.schema_migrations (version) FROM stdin;
2009  20171026211738
2010  20171026211808
2011  20171026211834
2012  20180103212743
2013  20180108183307
2014  20180119214651
2015  20180125194653
2016  00
2017  20210710035447
2018  20210722035447
2019  20210730183235
2020  20210909172000
2021  20210927181326
2022  20211122151130
2023  20211124214934
2024  20211202183645
2025  20220114185221
2026  20220114185340
2027  20220224000811
2028  20220323170000
2029  20220429102000
2030  \.
2031
2032
2033  --
2034  -- Data for Name: users; Type: TABLE DATA; Schema: auth; Owner: supabase_auth_admin
```

```
--

COPY auth.users (instance_id, id, aud, role, email, encrypted_password, email_confirmed_at, invited_at, confirmation_token, confirmation_sent_at, recovery_token, recovery_sent_at, email_change_token_new, email_change, email_change_sent_at, last_sign_in_at, raw_app_meta_data, raw_user_meta_data, is_super_admin, created_at, updated_at, phone, phone_confirmed_at, phone_change, phone_change_token, phone_change_sent_at, email_change_token_current, email_change_confirm_status, banned_until, reauthentication_token, reauthentication_sent_at) FROM stdin;
\.


--
-- Data for Name: Decisions; Type: TABLE DATA; Schema: public; Owner: supabase_admin
--

COPY public."Decisions" (id, project_id, voting_result, is_project_accepted, date_time_timezone) FROM stdin;
4	4	66.67	t	2022-04-27 08:11:43+00
1	2	66.67	t	2022-05-05 09:12:01+00
2	3	100	t	2022-04-24 19:09:45+00
3	5	100	t	2022-04-24 19:12:23+00
5	9	100	t	2022-05-12 11:42:13+00
6	6	100	t	2022-05-11 11:42:51+00
7	7	100	t	2022-05-11 02:47:36+00
8	8	100	t	2022-05-16 11:49:17+00
9	1	0	f	2022-05-06 05:50:09+00
\.


--
-- Data for Name: Donations; Type: TABLE DATA; Schema: public; Owner: supabase_admin
--

COPY public."Donations" (id, donor_id, project_id, proof_of_completion_file, project_completion_date_time_timezone) FROM stdin;
1	1	5	\N	\N
\.


--
-- Data for Name: Donors; Type: TABLE DATA; Schema: public; Owner: supabase_admin
--

COPY public."Donors" (id, wallet_address) FROM stdin;
1	0.0.34735190
\.


--
-- Data for Name: Governors; Type: TABLE DATA; Schema: public; Owner: supabase_admin
--

COPY public."Governors" (id, wallet_address) FROM stdin;
1	0.0.34735139
2	0.0.34735159
3	0.0.34735165
\.


--
-- Data for Name: Organizations; Type: TABLE DATA; Schema: public; Owner: supabase_admin
--
```

```
COPY public."Organizations" (id, name, website, country, active, other_info, image_file, wallet_address) FROM stdin;
1	Water From Air Foundation	https://waterfromair.org.tr	Turkey	t	Lack of access to water is the root of many economic and social problems around the world. Our charity tackles and aims to resolve this problems for communities around the world.	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/R001.jpg	0.0.34735175
2	Boon Lay Goodness Movement	https://boonlaygoodness.com	Singapore	t	Our charity organization is to reduce all types of suffering for people around the world. Dont eye power. Instead, join us and contribute.	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/R002.jpg	0.0.34735181
3	Lagos United Charity	https://lagosunitedcharity.org	Nigeria	t	Lagos United Charity aims to alleviate poverty in Lagos and its suburbs. Rather than coordinating direct food donations, our goal is to provide the tools to reduce poverty.	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/R003.jpg	0.0.34735186
\.


--
-- Data for Name: Projects; Type: TABLE DATA; Schema: public; Owner: supabase_admin
--

COPY public."Projects" (id, solution_id, organization_id, budget_usd, other_info, country, project_duration_days, status, "date_time_timezone ", "mintPriceHBAR", "maxNFTSupply") FROM stdin;
1	1	1	2500	\N	Indonesia	35	Accepted	2022-05-03 18:20:23+00	1000	27
3	1	1	3000	\N	Peru	30	Accepted	2022-05-02 02:24:42+00	2000	16
4	2	2	3000	\N	Bangladesh	45	Accepted	2022-04-26 17:25:03+00	2000	16
6	1	1	3500	\N	India	60	Accepted	2022-05-10 08:22:05+00	2000	19
7	2	2	3200	\N	Malaysia	50	Accepted	2022-05-10 10:20:18+00	5000	7
8	3	2	1600	\N	Singapore	23	Accepted	2022-05-13 15:00:17+00	1000	18
5	7	3	2800	\N	Nigeria	55	Initiated	2022-04-22 16:24:00+00	2000	15
2	4	2	2500	\N	Vietnam	40	Accepted	2022-05-08 16:21:24+00	1000	27
9	9	3	2000	\N	India	20	Accepted	2022-05-11 19:33:54+00	2000	11
\.


--
-- Data for Name: SingleVotes; Type: TABLE DATA; Schema: public; Owner: supabase_admin
--

COPY public."SingleVotes" (id, governor_id, vote_value, project_id, date_time_timezone) FROM stdin;
1	1	0	1	2022-05-07 19:00:50+00
5	3	1	2	2022-05-01 12:04:35+00
6	2	1	3	2022-05-02 08:01:46+00
7	3	1	3	2022-05-02 08:03:33+00
8	1	1	4	2022-04-24 11:06:37+00
3	1	1	2	2022-05-01 08:59:16+00
9	2	1	4	2022-04-24 10:08:20+00
4	2	0	2	2022-05-01 19:00:58+00
10	3	0	4	2022-04-24 19:03:41+00
11	1	1	5	2022-04-21 19:05:37+00
12	2	1	5	2022-04-20 19:06:07+00
2	2	0	1	2022-05-04 08:00:20+00
13	3	1	9	2022-05-12 19:35:49+00
14	2	1	6	2022-05-10 05:42:00+00
15	1	1	7	2022-05-10 12:43:30+00
16	3	1	7	2022-05-10 16:44:12+00
17	1	1	8	2022-05-14 02:48:47+00
\.


--
-- Data for Name: Solutions; Type: TABLE DATA; Schema: public; Owner: supabase_admin
```

```
--

COPY public."Solutions" (id, name, category, nature, summary, estimated_cost_usd, cost_breakdown, information_source, application_scenarios, applicable_in_city, applicable_in_nature, low_tech, simplicity, low_cost, portability, versatility, local_materials_required, labour_hours_required, product_company_name, product_webpage, other_info, applied_by_organizations, image_file_name) FROM stdin;
7	ecoDrone	Soil	Generating	UAV (Unmanned Aerial Vehicle) that can drop 2500 seedbombs in 10 minutes	3200	2200 USD for the drone, 1000 USD for seed bombs	https://youtu.be/byFrZ1T3EHI	Empty land with reasonable rainfall, which still has soil with enough quality to grow plants and trees	f	t	5	6	6	7	7	\N	5	Ecording	https://ecording.org/en/	https://ecording.org/en/faq/	Ecording	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/S007.jpg?t=2022-05-14T17:40:19.655Z
2	HyPump	Water	Transporting	Hydro-powered pump that can pump water to farmers over a mile away, without requiring any electricity for operation	1500	Estimated value of one pump from the manufacturer is 1500 USD, including worldwide shipment.	https://youtu.be/lPDvjIk5cQY	Built on rivers and streams in areas where there is no access to electricity.	f	t	6	7	8	7	7	Pickaxe, shovel, wheelbarrow	50	\N	\N	https://scalingfrontierinnovation.org/spotlights/aqysta-barsha-pump/	aQysta	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/S002.jpg
5	WaterLily	Energy	Generating	Generating energy for USB compatible devices	400	Turbine is 190 USD per piece, plus 10 USD for cable, 100 USD for wind kit, 100 USD for tripod	https://youtu.be/rxeMw1paqRI	Windy locations or flowing waters, best for people staying and moving in nature	t	t	7	8	8	9	6	\N	1	WaterLily	https://www.waterlilyturbine.com/	\N	\N	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/S005.jpg?t=2022-05-14T17:40:02.621Z
1	Fog catchers	Water	Generating	Water generation using nets in misty locations	2300	100 USD for each net, 15 nets minimum, giving 1500 USD for nets. Then another 750 USD for water tank and piping. 2250 USD total.	https://youtu.be/YxRONAZoMDk	Higher elevated remote locations with myst, or very mysty locations	f	t	9	8	7	7	7	Wooden poles and nets, hammer and nails, hand tools for piping	100	\N	\N	\N	\N	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/S001.jpg\r\n
3	Water-Gate	Water	Controlling	Inflatable portable mini-dam that stops floods and other excess flow of water	2000	2000 USD for each Water-Gate portable dam	https://youtu.be/arZbmV9vqug	Positioned on rivers, water currents, lake sides, and dams, when there is excess water flow	t	t	8	7	7	7	8	\N	3	MegaSecur	https://water-gate.com/en/barriers/see-all-barriers/	https://megasecur.com/en/services/advisory-team/	\N	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/S003.jpg?t=2022-05-14T12:56:53.987Z
4	Warka Water	Water	Generating	Bamboo tower, with mesh net inside, to cultivate water from fog or rain water	1500	1450 USD for the mesh net that collects the water, 50 USD for ropes	https://youtu.be/U3npQeku4zA	Humid foggy remote locations that do not have reliable water source but humidity in the air	f	t	8	8	8	5	6	Local Bamboo	30	Warka Water	https://www.warkawater.org/\n	\N	Warka Water	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/S004.jpg?t=2022-05-14T13:10:10.091Z
6	Solar Cooker	Energy	Generating	Cooker that harvests solar energy into heat	500	500 USD for each cooker	https://youtu.be/7vvCF3TZ9GE	Locations exposed to sunlight	t	t	9	9	9	7	9	Screwdriver, fastener	3	Solar Chef	https://solar-chef.com/	https://www.solarcookers.org/resources/download	Solar Cookers International	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/S006.jpg?t=2022-05-14T17:40:15.330Z
8	Water-Saving Nozzle	Water	Saving	Sink faucet nozzle that saves up to 98 percent water	1000	50 USD per nozzle, set of 20 nozzles for a building or facility	https://youtu.be/AWRxhbKWvZk	Reliable water piping system with minimum water flow pressure of 3 BAR	t	f	9	9	9	10	9	\N	1	Altered	https://alteredcompany.com/collections/nozzles	https://alteredcompany.com/collections/nozzles	Altered	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/S008.jpg?t=2022-05-14T19:27:27.905Z
9	Clay Refrigerator	Food	Storing	Zero-electricity clay refrigerator for preserving food and medicine	1500	150 USD for each clay refrigerator, set of 10 refrigerators for poor neighborhoods	https://youtu.be/WPYzV64dUuU	Remote areas with no electricity or poor neighborhoods, in relatively hot regions	t	t	9	9	8
```

```
6	8	\N	2	Mitti Cool	https://mitticool.com/product/mitticool-clay-refrigerator50-liter/	https://mitticool.com/product/mitticool-clay-refrigerator50-liter/	Mitti Cool	https://epvzhrganrzfwexmiksg.supabase.co/storage/v1/object/public/images/S009.jpg?t=2022-05-14T19:27:32.635Z
\.


--
-- Data for Name: SystemConstants; Type: TABLE DATA; Schema: public; Owner: supabase_admin
--

COPY public."SystemConstants" (id, min_perc_vote, path_reward_per_vote, hours_to_vote, days_to_claim_donation, days_to_complete_project) FROM stdin;
1	70	1	120	7	120
\.


--
-- Data for Name: schema_migrations; Type: TABLE DATA; Schema: realtime; Owner: supabase_admin
--

COPY realtime.schema_migrations (version, inserted_at) FROM stdin;
20211116024918	2022-05-03 08:35:23
20211116045059	2022-05-03 08:35:23
20211116050929	2022-05-03 08:35:23
20211116051442	2022-05-03 08:35:23
20211116212300	2022-05-03 08:35:23
20211116213355	2022-05-03 08:35:23
20211116213934	2022-05-03 08:35:23
20211116214523	2022-05-03 08:35:23
20211122062447	2022-05-03 08:35:23
20211124070109	2022-05-03 08:35:23
20211202204204	2022-05-03 08:35:23
20211202204605	2022-05-03 08:35:23
20211210212804	2022-05-03 08:35:23
20211228014915	2022-05-03 08:35:23
20220107221237	2022-05-03 08:35:23
20220228202821	2022-05-03 08:35:23
20220312004840	2022-05-03 08:35:23
\.


--
-- Data for Name: subscription; Type: TABLE DATA; Schema: realtime; Owner: supabase_admin
--

COPY realtime.subscription (id, subscription_id, entity, filters, claims, created_at) FROM stdin;
\.


--
-- Data for Name: buckets; Type: TABLE DATA; Schema: storage; Owner: supabase_storage_admin
--

COPY storage.buckets (id, name, owner, created_at, updated_at, public) FROM stdin;
images	images	\N	2022-05-10 11:21:08.894454+00	2022-05-10 11:21:08.894454+00	t
\.


--
```

```
-- Data for Name: migrations; Type: TABLE DATA; Schema: storage; Owner: supabase_storage_admin
--

COPY storage.migrations (id, name, hash, executed_at) FROM stdin;
0	create-migrations-table	e18db593bcde2aca2a408c4d1100f6abba2195df	2022-05-03 08:35:17.497422
1	initialmigration	6ab16121fbaa08bbd11b712d05f358f9b555d777	2022-05-03 08:35:17.502649
2	pathtoken-column	49756be03be4c17bb85fe70d4a861f27de7e49ad	2022-05-03 08:35:17.506822
3	add-migrations-rls	bb5d124c53d68635a883e399426c6a5a25fc893d	2022-05-03 08:35:17.530562
4	add-size-functions	6d79007d04f5acd288c9c250c42d2d5fd286c54d	2022-05-03 08:35:17.534984
5	change-column-name-in-get-size	fd65688505d2ffa9fbdc58a944348dd8604d688c	2022-05-03 08:35:17.540145
6	add-rls-to-buckets	63e2bab75a2040fee8e3fb3f15a0d26f3380e9b6	2022-05-03 08:35:17.546741
7	add-public-to-buckets	82568934f8a4d9e0a85f126f6fb483ad8214c418	2022-05-03 08:35:17.551492
8	fix-search-function	1a43a40eddb525f2e2f26efd709e6c06e58e059c	2022-05-03 08:35:17.557016
9	search-files-search-function	34c096597eb8b9d077fdfdde9878c88501b2fafc	2022-05-03 08:35:17.562544
\.


--
-- Data for Name: objects; Type: TABLE DATA; Schema: storage; Owner: supabase_storage_admin
--

COPY storage.objects (id, bucket_id, name, owner, created_at, updated_at, last_accessed_at, metadata) FROM stdin;
1a24f118-d8cf-44c4-8820-196c45a33bd4	images	.emptyFolderPlaceholder	\N	2022-05-10 11:23:17.18104+00	2022-05-10 11:23:17.18104+00	2022-05-10 11:23:17.18104+00	{"size": 0, "mimetype": "application/octet-stream", "cacheControl": "max-age=3600"}
1474c152-9408-4304-a867-790ace7f11ed	images	R001.jpg	\N	2022-05-10 13:13:19.084186+00	2022-05-10 13:13:19.084186+00	2022-05-10 13:13:19.084186+00	{"size": 237753, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
f019f347-4d36-475e-ba5e-3d181ea9f3d9	images	R002.jpg	\N	2022-05-10 13:13:19.130669+00	2022-05-10 13:13:19.130669+00	2022-05-10 13:13:19.130669+00	{"size": 331528, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
862af812-ecb8-4f77-8c22-029a2bced9db	images	R003.jpg	\N	2022-05-10 13:13:19.259245+00	2022-05-10 13:13:19.259245+00	2022-05-10 13:13:19.259245+00	{"size": 635309, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
87136908-0fc3-4014-a087-51003199bb25	images	S001.jpg	\N	2022-05-10 13:13:19.231481+00	2022-05-10 13:13:36.289+00	{"size": 703551, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
50a42f99-a548-4732-b2c8-bf55239dc5fa	images	S002.jpg	\N	2022-05-10 13:13:19.221684+00	2022-05-10 13:13:43.774+00	{"size": 622772, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
17d6445a-bf45-4d36-9e33-d8a4ceb54d2f	images	S003.jpg	\N	2022-05-10 13:13:19.591688+00	2022-05-10 13:13:50.119+00	{"size": 360941, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
0f693c7b-6377-4932-a9d7-38e64753124e	images	S004.jpg	\N	2022-05-10 13:13:19.630207+00	2022-05-10 13:13:55.81+00	{"size": 286014, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
df408a4d-17f9-42ad-b174-3e28ab04384e	images	S005.jpg	\N	2022-05-14 17:39:18.373798+00	2022-05-14 17:39:46.911+00	{"size": 428782, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
2f86c7eb-17e7-4c21-b469-7b04809089fc	images	S006.jpg	\N	2022-05-14 17:39:26.107618+00	2022-05-14 17:39:54.668+00	{"size":
```

```
388363, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
e0c5d966-41df-46ab-ad51-0f1574e8d1b1    images  S007.jpg    \N  2022-05-14 17:39:25.711855+00   2022-05-14 17:39:25.711855+00   2022-05-14 17:40:01.348+00  {"size": 149362, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
e6e5cd24-d5ae-4104-b489-10b1ac6d6575    images  S008.jpg    \N  2022-05-14 19:26:52.608695+00   2022-05-14 19:26:52.608695+00   2022-05-14 19:27:12.05+00   {"size": 432611, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
8307ed92-553e-45d2-b137-02871ba93f9b    images  S009.jpg    \N  2022-05-14 19:26:52.328964+00   2022-05-14 19:26:52.328964+00   2022-05-14 19:27:26.024+00  {"size": 186799, "mimetype": "image/jpeg", "cacheControl": "max-age=3600"}
\.


--
-- Name: refresh_tokens_id_seq; Type: SEQUENCE SET; Schema: auth; Owner: supabase_auth_admin
--

SELECT pg_catalog.setval('auth.refresh_tokens_id_seq', 1, false);


--
-- Name: Decision_id_seq; Type: SEQUENCE SET; Schema: public; Owner: supabase_admin
--

SELECT pg_catalog.setval('public."Decision_id_seq"', 1, true);


--
-- Name: Donation_id_seq; Type: SEQUENCE SET; Schema: public; Owner: supabase_admin
--

SELECT pg_catalog.setval('public."Donation_id_seq"', 1, false);


--
-- Name: Donor_id_seq; Type: SEQUENCE SET; Schema: public; Owner: supabase_admin
--

SELECT pg_catalog.setval('public."Donor_id_seq"', 1, false);


--
-- Name: Governors_id_seq; Type: SEQUENCE SET; Schema: public; Owner: supabase_admin
--

SELECT pg_catalog.setval('public."Governors_id_seq"', 1, false);


--
-- Name: Organizations_id_seq; Type: SEQUENCE SET; Schema: public; Owner: supabase_admin
--

SELECT pg_catalog.setval('public."Organizations_id_seq"', 4, true);


--
-- Name: Projects_id_seq; Type: SEQUENCE SET; Schema: public; Owner: supabase_admin
--

SELECT pg_catalog.setval('public."Projects_id_seq"', 6, true);


--
```

```
--
-- Name: SingleVote_id_seq; Type: SEQUENCE SET; Schema: public; Owner: supabase_admin
--

SELECT pg_catalog.setval('public."SingleVote_id_seq"', 1, false);


--
-- Name: Solutions_id_seq; Type: SEQUENCE SET; Schema: public; Owner: supabase_admin
--

SELECT pg_catalog.setval('public."Solutions_id_seq"', 1, false);


--
-- Name: SystemConstants_id_seq; Type: SEQUENCE SET; Schema: public; Owner: supabase_admin
--

SELECT pg_catalog.setval('public."SystemConstants_id_seq"', 1, false);


--
-- Name: subscription_id_seq; Type: SEQUENCE SET; Schema: realtime; Owner: supabase_admin
--

SELECT pg_catalog.setval('realtime.subscription_id_seq', 1, false);


--
-- Name: audit_log_entries audit_log_entries_pkey; Type: CONSTRAINT; Schema: auth; Owner: supabase_auth_admin
--

ALTER TABLE ONLY auth.audit_log_entries
    ADD CONSTRAINT audit_log_entries_pkey PRIMARY KEY (id);


--
-- Name: identities identities_pkey; Type: CONSTRAINT; Schema: auth; Owner: supabase_auth_admin
--

ALTER TABLE ONLY auth.identities
    ADD CONSTRAINT identities_pkey PRIMARY KEY (provider, id);


--
-- Name: instances instances_pkey; Type: CONSTRAINT; Schema: auth; Owner: supabase_auth_admin
--

ALTER TABLE ONLY auth.instances
    ADD CONSTRAINT instances_pkey PRIMARY KEY (id);


--
-- Name: refresh_tokens refresh_tokens_pkey; Type: CONSTRAINT; Schema: auth; Owner: supabase_auth_admin
--

ALTER TABLE ONLY auth.refresh_tokens
    ADD CONSTRAINT refresh_tokens_pkey PRIMARY KEY (id);


```

```
--
-- Name: refresh_tokens refresh_tokens_token_unique; Type: CONSTRAINT; Schema: auth; Owner: supabase_auth_admin
--

ALTER TABLE ONLY auth.refresh_tokens
    ADD CONSTRAINT refresh_tokens_token_unique UNIQUE (token);


--
-- Name: schema_migrations schema_migrations_pkey; Type: CONSTRAINT; Schema: auth; Owner: supabase_auth_admin
--

ALTER TABLE ONLY auth.schema_migrations
    ADD CONSTRAINT schema_migrations_pkey PRIMARY KEY (version);


--
-- Name: users users_email_key; Type: CONSTRAINT; Schema: auth; Owner: supabase_auth_admin
--

ALTER TABLE ONLY auth.users
    ADD CONSTRAINT users_email_key UNIQUE (email);


--
-- Name: users users_phone_key; Type: CONSTRAINT; Schema: auth; Owner: supabase_auth_admin
--

ALTER TABLE ONLY auth.users
    ADD CONSTRAINT users_phone_key UNIQUE (phone);


--
-- Name: users users_pkey; Type: CONSTRAINT; Schema: auth; Owner: supabase_auth_admin
--

ALTER TABLE ONLY auth.users
    ADD CONSTRAINT users_pkey PRIMARY KEY (id);


--
-- Name: Decisions Decision_pkey; Type: CONSTRAINT; Schema: public; Owner: supabase_admin
--

ALTER TABLE ONLY public."Decisions"
    ADD CONSTRAINT "Decision_pkey" PRIMARY KEY (id);


--
-- Name: Donations Donation_pkey; Type: CONSTRAINT; Schema: public; Owner: supabase_admin
--

ALTER TABLE ONLY public."Donations"
    ADD CONSTRAINT "Donation_pkey" PRIMARY KEY (id);


--
-- Name: Donors Donor_pkey; Type: CONSTRAINT; Schema: public; Owner: supabase_admin
--
```

```
ALTER TABLE ONLY public."Donors"
    ADD CONSTRAINT "Donor_pkey" PRIMARY KEY (id);


--
-- Name: Governors Governors_pkey; Type: CONSTRAINT; Schema: public; Owner: supabase_admin
--

ALTER TABLE ONLY public."Governors"
    ADD CONSTRAINT "Governors_pkey" PRIMARY KEY (id);


--
-- Name: Organizations Organizations_pkey; Type: CONSTRAINT; Schema: public; Owner: supabase_admin
--

ALTER TABLE ONLY public."Organizations"
    ADD CONSTRAINT "Organizations_pkey" PRIMARY KEY (id);


--
-- Name: Projects Projects_pkey; Type: CONSTRAINT; Schema: public; Owner: supabase_admin
--

ALTER TABLE ONLY public."Projects"
    ADD CONSTRAINT "Projects_pkey" PRIMARY KEY (id);


--
-- Name: SingleVotes SingleVote_pkey; Type: CONSTRAINT; Schema: public; Owner: supabase_admin
--

ALTER TABLE ONLY public."SingleVotes"
    ADD CONSTRAINT "SingleVote_pkey" PRIMARY KEY (id);


--
-- Name: Solutions Solutions_pkey; Type: CONSTRAINT; Schema: public; Owner: supabase_admin
--

ALTER TABLE ONLY public."Solutions"
    ADD CONSTRAINT "Solutions_pkey" PRIMARY KEY (id);


--
-- Name: SystemConstants SystemConstants_pkey; Type: CONSTRAINT; Schema: public; Owner: supabase_admin
--

ALTER TABLE ONLY public."SystemConstants"
    ADD CONSTRAINT "SystemConstants_pkey" PRIMARY KEY (id);


--
-- Name: subscription pk_subscription; Type: CONSTRAINT; Schema: realtime; Owner: supabase_admin
--

ALTER TABLE ONLY realtime.subscription
    ADD CONSTRAINT pk_subscription PRIMARY KEY (id);
```

```
2474
2475
2476  --
2477  -- Name: schema_migrations schema_migrations_pkey; Type: CONSTRAINT; Schema: realtime;
      Owner: supabase_admin
2478  --
2479
2480  ALTER TABLE ONLY realtime.schema_migrations
2481      ADD CONSTRAINT schema_migrations_pkey PRIMARY KEY (version);
2482
2483
2484  --
2485  -- Name: buckets buckets_pkey; Type: CONSTRAINT; Schema: storage; Owner:
      supabase_storage_admin
2486  --
2487
2488  ALTER TABLE ONLY storage.buckets
2489      ADD CONSTRAINT buckets_pkey PRIMARY KEY (id);
2490
2491
2492  --
2493  -- Name: migrations migrations_name_key; Type: CONSTRAINT; Schema: storage; Owner:
      supabase_storage_admin
2494  --
2495
2496  ALTER TABLE ONLY storage.migrations
2497      ADD CONSTRAINT migrations_name_key UNIQUE (name);
2498
2499
2500  --
2501  -- Name: migrations migrations_pkey; Type: CONSTRAINT; Schema: storage; Owner:
      supabase_storage_admin
2502  --
2503
2504  ALTER TABLE ONLY storage.migrations
2505      ADD CONSTRAINT migrations_pkey PRIMARY KEY (id);
2506
2507
2508  --
2509  -- Name: objects objects_pkey; Type: CONSTRAINT; Schema: storage; Owner:
      supabase_storage_admin
2510  --
2511
2512  ALTER TABLE ONLY storage.objects
2513      ADD CONSTRAINT objects_pkey PRIMARY KEY (id);
2514
2515
2516  --
2517  -- Name: audit_logs_instance_id_idx; Type: INDEX; Schema: auth; Owner:
      supabase_auth_admin
2518  --
2519
2520  CREATE INDEX audit_logs_instance_id_idx ON auth.audit_log_entries USING btree (
      instance_id);
2521
2522
2523  --
2524  -- Name: confirmation_token_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
2525  --
2526
2527  CREATE UNIQUE INDEX confirmation_token_idx ON auth.users USING btree (confirmation_token
      ) WHERE ((confirmation_token)::text !~ '^[0-9 ]*$'::text);
2528
2529
```

```
--
-- Name: email_change_token_current_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
--

CREATE UNIQUE INDEX email_change_token_current_idx ON auth.users USING btree (email_change_token_current) WHERE ((email_change_token_current)::text !~ '^[0-9 ]*$'::text);


--
-- Name: email_change_token_new_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
--

CREATE UNIQUE INDEX email_change_token_new_idx ON auth.users USING btree (email_change_token_new) WHERE ((email_change_token_new)::text !~ '^[0-9 ]*$'::text);


--
-- Name: identities_user_id_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
--

CREATE INDEX identities_user_id_idx ON auth.identities USING btree (user_id);


--
-- Name: reauthentication_token_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
--

CREATE UNIQUE INDEX reauthentication_token_idx ON auth.users USING btree (reauthentication_token) WHERE ((reauthentication_token)::text !~ '^[0-9 ]*$'::text);


--
-- Name: recovery_token_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
--

CREATE UNIQUE INDEX recovery_token_idx ON auth.users USING btree (recovery_token) WHERE ((recovery_token)::text !~ '^[0-9 ]*$'::text);


--
-- Name: refresh_tokens_instance_id_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
--

CREATE INDEX refresh_tokens_instance_id_idx ON auth.refresh_tokens USING btree (instance_id);


--
-- Name: refresh_tokens_instance_id_user_id_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
--

CREATE INDEX refresh_tokens_instance_id_user_id_idx ON auth.refresh_tokens USING btree (instance_id, user_id);


--
-- Name: refresh_tokens_parent_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
--
```

```
2582
2583  CREATE INDEX refresh_tokens_parent_idx ON auth.refresh_tokens USING btree (parent);
2584
2585
2586  --
2587  -- Name: refresh_tokens_token_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
2588  --
2589
2590  CREATE INDEX refresh_tokens_token_idx ON auth.refresh_tokens USING btree (token);
2591
2592
2593  --
2594  -- Name: users_instance_id_email_idx; Type: INDEX; Schema: auth; Owner:
      supabase_auth_admin
2595  --
2596
2597  CREATE INDEX users_instance_id_email_idx ON auth.users USING btree (instance_id, lower((
      email)::text));
2598
2599
2600  --
2601  -- Name: users_instance_id_idx; Type: INDEX; Schema: auth; Owner: supabase_auth_admin
2602  --
2603
2604  CREATE INDEX users_instance_id_idx ON auth.users USING btree (instance_id);
2605
2606
2607  --
2608  -- Name: ix_realtime_subscription_entity; Type: INDEX; Schema: realtime; Owner:
      supabase_admin
2609  --
2610
2611  CREATE INDEX ix_realtime_subscription_entity ON realtime.subscription USING hash (entity
      );
2612
2613
2614  --
2615  -- Name: subscription_subscription_id_entity_filters_key; Type: INDEX; Schema:
      realtime; Owner: supabase_admin
2616  --
2617
2618  CREATE UNIQUE INDEX subscription_subscription_id_entity_filters_key ON realtime.
      subscription USING btree (subscription_id, entity, filters);
2619
2620
2621  --
2622  -- Name: bname; Type: INDEX; Schema: storage; Owner: supabase_storage_admin
2623  --
2624
2625  CREATE UNIQUE INDEX bname ON storage.buckets USING btree (name);
2626
2627
2628  --
2629  -- Name: bucketid_objname; Type: INDEX; Schema: storage; Owner: supabase_storage_admin
2630  --
2631
2632  CREATE UNIQUE INDEX bucketid_objname ON storage.objects USING btree (bucket_id, name);
2633
2634
2635  --
2636  -- Name: name_prefix_search; Type: INDEX; Schema: storage; Owner: supabase_storage_admin
2637  --
2638
2639  CREATE INDEX name_prefix_search ON storage.objects USING btree (name text_pattern_ops);
```

```
2640
2641
2642    --
2643    -- Name: subscription tr_check_filters; Type: TRIGGER; Schema: realtime; Owner:
        supabase_admin
2644    --
2645
2646    CREATE TRIGGER tr_check_filters BEFORE INSERT OR UPDATE ON realtime.subscription FOR
        EACH ROW EXECUTE FUNCTION realtime.subscription_check_filters();
2647
2648
2649    --
2650    -- Name: identities identities_user_id_fkey; Type: FK CONSTRAINT; Schema: auth; Owner:
        supabase_auth_admin
2651    --
2652
2653    ALTER TABLE ONLY auth.identities
2654        ADD CONSTRAINT identities_user_id_fkey FOREIGN KEY (user_id) REFERENCES auth.users(
            id) ON DELETE CASCADE;
2655
2656
2657    --
2658    -- Name: refresh_tokens refresh_tokens_parent_fkey; Type: FK CONSTRAINT; Schema: auth;
        Owner: supabase_auth_admin
2659    --
2660
2661    ALTER TABLE ONLY auth.refresh_tokens
2662        ADD CONSTRAINT refresh_tokens_parent_fkey FOREIGN KEY (parent) REFERENCES auth.
            refresh_tokens(token);
2663
2664
2665    --
2666    -- Name: Donations Donation_donor_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner:
        supabase_admin
2667    --
2668
2669    ALTER TABLE ONLY public."Donations"
2670        ADD CONSTRAINT "Donation_donor_id_fkey" FOREIGN KEY (donor_id) REFERENCES public.
            "Donors"(id);
2671
2672
2673    --
2674    -- Name: Donations Donation_project_id_fkey; Type: FK CONSTRAINT; Schema: public;
        Owner: supabase_admin
2675    --
2676
2677    ALTER TABLE ONLY public."Donations"
2678        ADD CONSTRAINT "Donation_project_id_fkey" FOREIGN KEY (project_id) REFERENCES public
            ."Projects"(id);
2679
2680
2681    --
2682    -- Name: Projects Projects_organization_id_fkey; Type: FK CONSTRAINT; Schema: public;
        Owner: supabase_admin
2683    --
2684
2685    ALTER TABLE ONLY public."Projects"
2686        ADD CONSTRAINT "Projects_organization_id_fkey" FOREIGN KEY (organization_id)
            REFERENCES public."Organizations"(id);
2687
2688
2689    --
2690    -- Name: Projects Projects_solution_id_fkey; Type: FK CONSTRAINT; Schema: public;
        Owner: supabase_admin
```

```
--

ALTER TABLE ONLY public."Projects"
    ADD CONSTRAINT "Projects_solution_id_fkey" FOREIGN KEY (solution_id) REFERENCES
    public."Solutions"(id);


--
-- Name: SingleVotes SingleVote_governor_id_fkey; Type: FK CONSTRAINT; Schema: public;
Owner: supabase_admin
--

ALTER TABLE ONLY public."SingleVotes"
    ADD CONSTRAINT "SingleVote_governor_id_fkey" FOREIGN KEY (governor_id) REFERENCES
    public."Governors"(id);


--
-- Name: SingleVotes SingleVotes_project_id_fkey; Type: FK CONSTRAINT; Schema: public;
Owner: supabase_admin
--

ALTER TABLE ONLY public."SingleVotes"
    ADD CONSTRAINT "SingleVotes_project_id_fkey" FOREIGN KEY (project_id) REFERENCES
    public."Projects"(id);


--
-- Name: buckets buckets_owner_fkey; Type: FK CONSTRAINT; Schema: storage; Owner:
supabase_storage_admin
--

ALTER TABLE ONLY storage.buckets
    ADD CONSTRAINT buckets_owner_fkey FOREIGN KEY (owner) REFERENCES auth.users(id);


--
-- Name: objects objects_bucketId_fkey; Type: FK CONSTRAINT; Schema: storage; Owner:
supabase_storage_admin
--

ALTER TABLE ONLY storage.objects
    ADD CONSTRAINT "objects_bucketId_fkey" FOREIGN KEY (bucket_id) REFERENCES storage.
    buckets(id);


--
-- Name: objects objects_owner_fkey; Type: FK CONSTRAINT; Schema: storage; Owner:
supabase_storage_admin
--

ALTER TABLE ONLY storage.objects
    ADD CONSTRAINT objects_owner_fkey FOREIGN KEY (owner) REFERENCES auth.users(id);


--
-- Name: buckets; Type: ROW SECURITY; Schema: storage; Owner: supabase_storage_admin
--

ALTER TABLE storage.buckets ENABLE ROW LEVEL SECURITY;

--
-- Name: migrations; Type: ROW SECURITY; Schema: storage; Owner: supabase_storage_admin
--
```

```
2746
2747  ALTER TABLE storage.migrations ENABLE ROW LEVEL SECURITY;
2748
2749  --
2750  -- Name: objects; Type: ROW SECURITY; Schema: storage; Owner: supabase_storage_admin
2751  --
2752
2753  ALTER TABLE storage.objects ENABLE ROW LEVEL SECURITY;
2754
2755  --
2756  -- Name: supabase_realtime; Type: PUBLICATION; Schema: -; Owner: postgres
2757  --
2758
2759  CREATE PUBLICATION supabase_realtime WITH (publish = 'insert, update, delete, truncate');
2760
2761
2762  ALTER PUBLICATION supabase_realtime OWNER TO postgres;
2763
2764  --
2765  -- Name: SCHEMA auth; Type: ACL; Schema: -; Owner: supabase_admin
2766  --
2767
2768  GRANT USAGE ON SCHEMA auth TO anon;
2769  GRANT USAGE ON SCHEMA auth TO authenticated;
2770  GRANT USAGE ON SCHEMA auth TO service_role;
2771  GRANT ALL ON SCHEMA auth TO supabase_auth_admin;
2772  GRANT ALL ON SCHEMA auth TO dashboard_user;
2773  GRANT ALL ON SCHEMA auth TO postgres;
2774
2775
2776  --
2777  -- Name: SCHEMA extensions; Type: ACL; Schema: -; Owner: postgres
2778  --
2779
2780  GRANT USAGE ON SCHEMA extensions TO anon;
2781  GRANT USAGE ON SCHEMA extensions TO authenticated;
2782  GRANT USAGE ON SCHEMA extensions TO service_role;
2783  GRANT ALL ON SCHEMA extensions TO dashboard_user;
2784
2785
2786  --
2787  -- Name: SCHEMA public; Type: ACL; Schema: -; Owner: postgres
2788  --
2789
2790  GRANT USAGE ON SCHEMA public TO anon;
2791  GRANT USAGE ON SCHEMA public TO authenticated;
2792  GRANT USAGE ON SCHEMA public TO service_role;
2793
2794
2795  --
2796  -- Name: SCHEMA graphql_public; Type: ACL; Schema: -; Owner: supabase_admin
2797  --
2798
2799  GRANT USAGE ON SCHEMA graphql_public TO postgres;
2800  GRANT USAGE ON SCHEMA graphql_public TO anon;
2801  GRANT USAGE ON SCHEMA graphql_public TO authenticated;
2802  GRANT USAGE ON SCHEMA graphql_public TO service_role;
2803
2804
2805  --
2806  -- Name: SCHEMA realtime; Type: ACL; Schema: -; Owner: supabase_admin
2807  --
2808
2809  GRANT USAGE ON SCHEMA realtime TO postgres;
```

```
2810
2811
2812    --
2813    -- Name: SCHEMA storage; Type: ACL; Schema: -; Owner: supabase_admin
2814    --
2815
2816    GRANT ALL ON SCHEMA storage TO postgres;
2817    GRANT USAGE ON SCHEMA storage TO anon;
2818    GRANT USAGE ON SCHEMA storage TO authenticated;
2819    GRANT USAGE ON SCHEMA storage TO service_role;
2820    GRANT ALL ON SCHEMA storage TO supabase_storage_admin;
2821    GRANT ALL ON SCHEMA storage TO dashboard_user;
2822
2823
2824    --
2825    -- Name: FUNCTION email(); Type: ACL; Schema: auth; Owner: supabase_auth_admin
2826    --
2827
2828    GRANT ALL ON FUNCTION auth.email() TO dashboard_user;
2829
2830
2831    --
2832    -- Name: FUNCTION role(); Type: ACL; Schema: auth; Owner: supabase_auth_admin
2833    --
2834
2835    GRANT ALL ON FUNCTION auth.role() TO dashboard_user;
2836
2837
2838    --
2839    -- Name: FUNCTION uid(); Type: ACL; Schema: auth; Owner: supabase_auth_admin
2840    --
2841
2842    GRANT ALL ON FUNCTION auth.uid() TO dashboard_user;
2843
2844
2845    --
2846    -- Name: FUNCTION algorithm_sign(signables text, secret text, algorithm text); Type:
        ACL; Schema: extensions; Owner: postgres
2847    --
2848
2849    GRANT ALL ON FUNCTION extensions.algorithm_sign(signables text, secret text, algorithm
        text) TO dashboard_user;
2850
2851
2852    --
2853    -- Name: FUNCTION armor(bytea); Type: ACL; Schema: extensions; Owner: postgres
2854    --
2855
2856    GRANT ALL ON FUNCTION extensions.armor(bytea) TO dashboard_user;
2857
2858
2859    --
2860    -- Name: FUNCTION armor(bytea, text[], text[]); Type: ACL; Schema: extensions; Owner:
        postgres
2861    --
2862
2863    GRANT ALL ON FUNCTION extensions.armor(bytea, text[], text[]) TO dashboard_user;
2864
2865
2866    --
2867    -- Name: FUNCTION crypt(text, text); Type: ACL; Schema: extensions; Owner: postgres
2868    --
2869
2870    GRANT ALL ON FUNCTION extensions.crypt(text, text) TO dashboard_user;
```

```
2871
2872
2873    --
2874    -- Name: FUNCTION dearmor(text); Type: ACL; Schema: extensions; Owner: postgres
2875    --
2876
2877    GRANT ALL ON FUNCTION extensions.dearmor(text) TO dashboard_user;
2878
2879
2880    --
2881    -- Name: FUNCTION decrypt(bytea, bytea, text); Type: ACL; Schema: extensions; Owner:
        postgres
2882    --
2883
2884    GRANT ALL ON FUNCTION extensions.decrypt(bytea, bytea, text) TO dashboard_user;
2885
2886
2887    --
2888    -- Name: FUNCTION decrypt_iv(bytea, bytea, bytea, text); Type: ACL; Schema: extensions;
        Owner: postgres
2889    --
2890
2891    GRANT ALL ON FUNCTION extensions.decrypt_iv(bytea, bytea, bytea, text) TO dashboard_user;
2892
2893
2894    --
2895    -- Name: FUNCTION digest(bytea, text); Type: ACL; Schema: extensions; Owner: postgres
2896    --
2897
2898    GRANT ALL ON FUNCTION extensions.digest(bytea, text) TO dashboard_user;
2899
2900
2901    --
2902    -- Name: FUNCTION digest(text, text); Type: ACL; Schema: extensions; Owner: postgres
2903    --
2904
2905    GRANT ALL ON FUNCTION extensions.digest(text, text) TO dashboard_user;
2906
2907
2908    --
2909    -- Name: FUNCTION encrypt(bytea, bytea, text); Type: ACL; Schema: extensions; Owner:
        postgres
2910    --
2911
2912    GRANT ALL ON FUNCTION extensions.encrypt(bytea, bytea, text) TO dashboard_user;
2913
2914
2915    --
2916    -- Name: FUNCTION encrypt_iv(bytea, bytea, bytea, text); Type: ACL; Schema: extensions;
        Owner: postgres
2917    --
2918
2919    GRANT ALL ON FUNCTION extensions.encrypt_iv(bytea, bytea, bytea, text) TO dashboard_user;
2920
2921
2922    --
2923    -- Name: FUNCTION gen_random_bytes(integer); Type: ACL; Schema: extensions; Owner:
        postgres
2924    --
2925
2926    GRANT ALL ON FUNCTION extensions.gen_random_bytes(integer) TO dashboard_user;
2927
2928
2929    --
```

```
2930  -- Name: FUNCTION gen_random_uuid(); Type: ACL; Schema: extensions; Owner: postgres
2931  --
2932
2933  GRANT ALL ON FUNCTION extensions.gen_random_uuid() TO dashboard_user;
2934
2935
2936  --
2937  -- Name: FUNCTION gen_salt(text); Type: ACL; Schema: extensions; Owner: postgres
2938  --
2939
2940  GRANT ALL ON FUNCTION extensions.gen_salt(text) TO dashboard_user;
2941
2942
2943  --
2944  -- Name: FUNCTION gen_salt(text, integer); Type: ACL; Schema: extensions; Owner: postgres
2945  --
2946
2947  GRANT ALL ON FUNCTION extensions.gen_salt(text, integer) TO dashboard_user;
2948
2949
2950  --
2951  -- Name: FUNCTION grant_pg_cron_access(); Type: ACL; Schema: extensions; Owner: postgres
2952  --
2953
2954  GRANT ALL ON FUNCTION extensions.grant_pg_cron_access() TO dashboard_user;
2955
2956
2957  --
2958  -- Name: FUNCTION grant_pg_net_access(); Type: ACL; Schema: extensions; Owner: postgres
2959  --
2960
2961  GRANT ALL ON FUNCTION extensions.grant_pg_net_access() TO dashboard_user;
2962
2963
2964  --
2965  -- Name: FUNCTION hmac(bytea, bytea, text); Type: ACL; Schema: extensions; Owner:
      postgres
2966  --
2967
2968  GRANT ALL ON FUNCTION extensions.hmac(bytea, bytea, text) TO dashboard_user;
2969
2970
2971  --
2972  -- Name: FUNCTION hmac(text, text, text); Type: ACL; Schema: extensions; Owner: postgres
2973  --
2974
2975  GRANT ALL ON FUNCTION extensions.hmac(text, text, text) TO dashboard_user;
2976
2977
2978  --
2979  -- Name: FUNCTION pg_stat_statements(showtext boolean, OUT userid oid, OUT dbid oid,
      OUT toplevel boolean, OUT queryid bigint, OUT query text, OUT plans bigint, OUT
      total_plan_time double precision, OUT min_plan_time double precision, OUT max_plan_time
      double precision, OUT mean_plan_time double precision, OUT stddev_plan_time double
      precision, OUT calls bigint, OUT total_exec_time double precision, OUT min_exec_time
      double precision, OUT max_exec_time double precision, OUT mean_exec_time double
      precision, OUT stddev_exec_time double precision, OUT rows bigint, OUT shared_blks_hit
      bigint, OUT shared_blks_read bigint, OUT shared_blks_dirtied bigint, OUT
      shared_blks_written bigint, OUT local_blks_hit bigint, OUT local_blks_read bigint, OUT
      local_blks_dirtied bigint, OUT local_blks_written bigint, OUT temp_blks_read bigint,
      OUT temp_blks_written bigint, OUT blk_read_time double precision, OUT blk_write_time
      double precision, OUT wal_records bigint, OUT wal_fpi bigint, OUT wal_bytes numeric);
      Type: ACL; Schema: extensions; Owner: postgres
2980  --
```

```
2981
2982  GRANT ALL ON FUNCTION extensions.pg_stat_statements(showtext boolean, OUT userid oid,
      OUT dbid oid, OUT toplevel boolean, OUT queryid bigint, OUT query text, OUT plans bigint
      , OUT total_plan_time double precision, OUT min_plan_time double precision, OUT
      max_plan_time double precision, OUT mean_plan_time double precision, OUT
      stddev_plan_time double precision, OUT calls bigint, OUT total_exec_time double
      precision, OUT min_exec_time double precision, OUT max_exec_time double precision, OUT
      mean_exec_time double precision, OUT stddev_exec_time double precision, OUT rows bigint,
       OUT shared_blks_hit bigint, OUT shared_blks_read bigint, OUT shared_blks_dirtied bigint
      , OUT shared_blks_written bigint, OUT local_blks_hit bigint, OUT local_blks_read bigint,
       OUT local_blks_dirtied bigint, OUT local_blks_written bigint, OUT temp_blks_read bigint
      , OUT temp_blks_written bigint, OUT blk_read_time double precision, OUT blk_write_time
      double precision, OUT wal_records bigint, OUT wal_fpi bigint, OUT wal_bytes numeric) TO
      dashboard_user;
2983
2984
2985  --
2986  -- Name: FUNCTION pg_stat_statements_info(OUT dealloc bigint, OUT stats_reset timestamp
      with time zone); Type: ACL; Schema: extensions; Owner: postgres
2987  --
2988
2989  GRANT ALL ON FUNCTION extensions.pg_stat_statements_info(OUT dealloc bigint, OUT
      stats_reset timestamp with time zone) TO dashboard_user;
2990
2991
2992  --
2993  -- Name: FUNCTION pg_stat_statements_reset(userid oid, dbid oid, queryid bigint); Type:
      ACL; Schema: extensions; Owner: postgres
2994  --
2995
2996  GRANT ALL ON FUNCTION extensions.pg_stat_statements_reset(userid oid, dbid oid, queryid
      bigint) TO dashboard_user;
2997
2998
2999  --
3000  -- Name: FUNCTION pgp_armor_headers(text, OUT key text, OUT value text); Type: ACL;
      Schema: extensions; Owner: postgres
3001  --
3002
3003  GRANT ALL ON FUNCTION extensions.pgp_armor_headers(text, OUT key text, OUT value text)
      TO dashboard_user;
3004
3005
3006  --
3007  -- Name: FUNCTION pgp_key_id(bytea); Type: ACL; Schema: extensions; Owner: postgres
3008  --
3009
3010  GRANT ALL ON FUNCTION extensions.pgp_key_id(bytea) TO dashboard_user;
3011
3012
3013  --
3014  -- Name: FUNCTION pgp_pub_decrypt(bytea, bytea); Type: ACL; Schema: extensions; Owner:
      postgres
3015  --
3016
3017  GRANT ALL ON FUNCTION extensions.pgp_pub_decrypt(bytea, bytea) TO dashboard_user;
3018
3019
3020  --
3021  -- Name: FUNCTION pgp_pub_decrypt(bytea, bytea, text); Type: ACL; Schema: extensions;
      Owner: postgres
3022  --
3023
3024  GRANT ALL ON FUNCTION extensions.pgp_pub_decrypt(bytea, bytea, text) TO dashboard_user;
```

```
3025
3026
3027    --
3028    -- Name: FUNCTION pgp_pub_decrypt(bytea, bytea, text, text); Type: ACL; Schema:
        extensions; Owner: postgres
3029    --
3030
3031    GRANT ALL ON FUNCTION extensions.pgp_pub_decrypt(bytea, bytea, text, text) TO
        dashboard_user;
3032
3033
3034    --
3035    -- Name: FUNCTION pgp_pub_decrypt_bytea(bytea, bytea); Type: ACL; Schema: extensions;
        Owner: postgres
3036    --
3037
3038    GRANT ALL ON FUNCTION extensions.pgp_pub_decrypt_bytea(bytea, bytea) TO dashboard_user;
3039
3040
3041    --
3042    -- Name: FUNCTION pgp_pub_decrypt_bytea(bytea, bytea, text); Type: ACL; Schema:
        extensions; Owner: postgres
3043    --
3044
3045    GRANT ALL ON FUNCTION extensions.pgp_pub_decrypt_bytea(bytea, bytea, text) TO
        dashboard_user;
3046
3047
3048    --
3049    -- Name: FUNCTION pgp_pub_decrypt_bytea(bytea, bytea, text, text); Type: ACL; Schema:
        extensions; Owner: postgres
3050    --
3051
3052    GRANT ALL ON FUNCTION extensions.pgp_pub_decrypt_bytea(bytea, bytea, text, text) TO
        dashboard_user;
3053
3054
3055    --
3056    -- Name: FUNCTION pgp_pub_encrypt(text, bytea); Type: ACL; Schema: extensions; Owner:
        postgres
3057    --
3058
3059    GRANT ALL ON FUNCTION extensions.pgp_pub_encrypt(text, bytea) TO dashboard_user;
3060
3061
3062    --
3063    -- Name: FUNCTION pgp_pub_encrypt(text, bytea, text); Type: ACL; Schema: extensions;
        Owner: postgres
3064    --
3065
3066    GRANT ALL ON FUNCTION extensions.pgp_pub_encrypt(text, bytea, text) TO dashboard_user;
3067
3068
3069    --
3070    -- Name: FUNCTION pgp_pub_encrypt_bytea(bytea, bytea); Type: ACL; Schema: extensions;
        Owner: postgres
3071    --
3072
3073    GRANT ALL ON FUNCTION extensions.pgp_pub_encrypt_bytea(bytea, bytea) TO dashboard_user;
3074
3075
3076    --
3077    -- Name: FUNCTION pgp_pub_encrypt_bytea(bytea, bytea, text); Type: ACL; Schema:
        extensions; Owner: postgres
```

```
--

GRANT ALL ON FUNCTION extensions.pgp_pub_encrypt_bytea(bytea, bytea, text) TO dashboard_user;


--
-- Name: FUNCTION pgp_sym_decrypt(bytea, text); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.pgp_sym_decrypt(bytea, text) TO dashboard_user;


--
-- Name: FUNCTION pgp_sym_decrypt(bytea, text, text); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.pgp_sym_decrypt(bytea, text, text) TO dashboard_user;


--
-- Name: FUNCTION pgp_sym_decrypt_bytea(bytea, text); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.pgp_sym_decrypt_bytea(bytea, text) TO dashboard_user;


--
-- Name: FUNCTION pgp_sym_decrypt_bytea(bytea, text, text); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.pgp_sym_decrypt_bytea(bytea, text, text) TO dashboard_user;


--
-- Name: FUNCTION pgp_sym_encrypt(text, text); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.pgp_sym_encrypt(text, text) TO dashboard_user;


--
-- Name: FUNCTION pgp_sym_encrypt(text, text, text); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.pgp_sym_encrypt(text, text, text) TO dashboard_user;


--
-- Name: FUNCTION pgp_sym_encrypt_bytea(bytea, text); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.pgp_sym_encrypt_bytea(bytea, text) TO dashboard_user;


--
```

```
3133  -- Name: FUNCTION pgp_sym_encrypt_bytea(bytea, text, text); Type: ACL; Schema:
      extensions; Owner: postgres
3134  --
3135
3136  GRANT ALL ON FUNCTION extensions.pgp_sym_encrypt_bytea(bytea, text, text) TO
      dashboard_user;
3137
3138
3139  --
3140  -- Name: FUNCTION sign(payload json, secret text, algorithm text); Type: ACL; Schema:
      extensions; Owner: postgres
3141  --
3142
3143  GRANT ALL ON FUNCTION extensions.sign(payload json, secret text, algorithm text) TO
      dashboard_user;
3144
3145
3146  --
3147  -- Name: FUNCTION try_cast_double(inp text); Type: ACL; Schema: extensions; Owner:
      postgres
3148  --
3149
3150  GRANT ALL ON FUNCTION extensions.try_cast_double(inp text) TO dashboard_user;
3151
3152
3153  --
3154  -- Name: FUNCTION url_decode(data text); Type: ACL; Schema: extensions; Owner: postgres
3155  --
3156
3157  GRANT ALL ON FUNCTION extensions.url_decode(data text) TO dashboard_user;
3158
3159
3160  --
3161  -- Name: FUNCTION url_encode(data bytea); Type: ACL; Schema: extensions; Owner: postgres
3162  --
3163
3164  GRANT ALL ON FUNCTION extensions.url_encode(data bytea) TO dashboard_user;
3165
3166
3167  --
3168  -- Name: FUNCTION uuid_generate_v1(); Type: ACL; Schema: extensions; Owner: postgres
3169  --
3170
3171  GRANT ALL ON FUNCTION extensions.uuid_generate_v1() TO dashboard_user;
3172
3173
3174  --
3175  -- Name: FUNCTION uuid_generate_v1mc(); Type: ACL; Schema: extensions; Owner: postgres
3176  --
3177
3178  GRANT ALL ON FUNCTION extensions.uuid_generate_v1mc() TO dashboard_user;
3179
3180
3181  --
3182  -- Name: FUNCTION uuid_generate_v3(namespace uuid, name text); Type: ACL; Schema:
      extensions; Owner: postgres
3183  --
3184
3185  GRANT ALL ON FUNCTION extensions.uuid_generate_v3(namespace uuid, name text) TO
      dashboard_user;
3186
3187
3188  --
3189  -- Name: FUNCTION uuid_generate_v4(); Type: ACL; Schema: extensions; Owner: postgres
```

```
--

GRANT ALL ON FUNCTION extensions.uuid_generate_v4() TO dashboard_user;


--
-- Name: FUNCTION uuid_generate_v5(namespace uuid, name text); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.uuid_generate_v5(namespace uuid, name text) TO dashboard_user;


--
-- Name: FUNCTION uuid_nil(); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.uuid_nil() TO dashboard_user;


--
-- Name: FUNCTION uuid_ns_dns(); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.uuid_ns_dns() TO dashboard_user;


--
-- Name: FUNCTION uuid_ns_oid(); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.uuid_ns_oid() TO dashboard_user;


--
-- Name: FUNCTION uuid_ns_url(); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.uuid_ns_url() TO dashboard_user;


--
-- Name: FUNCTION uuid_ns_x500(); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.uuid_ns_x500() TO dashboard_user;


--
-- Name: FUNCTION verify(token text, secret text, algorithm text); Type: ACL; Schema: extensions; Owner: postgres
--

GRANT ALL ON FUNCTION extensions.verify(token text, secret text, algorithm text) TO dashboard_user;


--
-- Name: FUNCTION rebuild_on_ddl(); Type: ACL; Schema: graphql; Owner: supabase_admin
--

GRANT ALL ON FUNCTION graphql.rebuild_on_ddl() TO postgres;
GRANT ALL ON FUNCTION graphql.rebuild_on_ddl() TO anon;
```

```
3250  GRANT ALL ON FUNCTION graphql.rebuild_on_ddl() TO authenticated;
3251  GRANT ALL ON FUNCTION graphql.rebuild_on_ddl() TO service_role;
3252
3253
3254  --
3255  -- Name: FUNCTION rebuild_on_drop(); Type: ACL; Schema: graphql; Owner: supabase_admin
3256  --
3257
3258  GRANT ALL ON FUNCTION graphql.rebuild_on_drop() TO postgres;
3259  GRANT ALL ON FUNCTION graphql.rebuild_on_drop() TO anon;
3260  GRANT ALL ON FUNCTION graphql.rebuild_on_drop() TO authenticated;
3261  GRANT ALL ON FUNCTION graphql.rebuild_on_drop() TO service_role;
3262
3263
3264  --
3265  -- Name: FUNCTION rebuild_schema(); Type: ACL; Schema: graphql; Owner: supabase_admin
3266  --
3267
3268  GRANT ALL ON FUNCTION graphql.rebuild_schema() TO postgres;
3269  GRANT ALL ON FUNCTION graphql.rebuild_schema() TO anon;
3270  GRANT ALL ON FUNCTION graphql.rebuild_schema() TO authenticated;
3271  GRANT ALL ON FUNCTION graphql.rebuild_schema() TO service_role;
3272
3273
3274  --
3275  -- Name: FUNCTION variable_definitions_sort(variable_definitions jsonb); Type: ACL;
     Schema: graphql; Owner: supabase_admin
3276  --
3277
3278  GRANT ALL ON FUNCTION graphql.variable_definitions_sort(variable_definitions jsonb) TO
     postgres;
3279  GRANT ALL ON FUNCTION graphql.variable_definitions_sort(variable_definitions jsonb) TO
     anon;
3280  GRANT ALL ON FUNCTION graphql.variable_definitions_sort(variable_definitions jsonb) TO
     authenticated;
3281  GRANT ALL ON FUNCTION graphql.variable_definitions_sort(variable_definitions jsonb) TO
     service_role;
3282
3283
3284  --
3285  -- Name: FUNCTION get_auth(p_usename text); Type: ACL; Schema: pgbouncer; Owner: postgres
3286  --
3287
3288  REVOKE ALL ON FUNCTION pgbouncer.get_auth(p_usename text) FROM PUBLIC;
3289  GRANT ALL ON FUNCTION pgbouncer.get_auth(p_usename text) TO pgbouncer;
3290
3291
3292  --
3293  -- Name: FUNCTION apply_rls(wal jsonb, max_record_bytes integer); Type: ACL; Schema:
     realtime; Owner: supabase_admin
3294  --
3295
3296  GRANT ALL ON FUNCTION realtime.apply_rls(wal jsonb, max_record_bytes integer) TO
     postgres;
3297  GRANT ALL ON FUNCTION realtime.apply_rls(wal jsonb, max_record_bytes integer) TO
     dashboard_user;
3298
3299
3300  --
3301  -- Name: FUNCTION build_prepared_statement_sql(prepared_statement_name text, entity
     regclass, columns realtime.wal_column[]); Type: ACL; Schema: realtime; Owner:
     supabase_admin
3302  --
3303
```

```
GRANT ALL ON FUNCTION realtime.build_prepared_statement_sql(prepared_statement_name text, entity regclass, columns realtime.wal_column[]) TO postgres;
GRANT ALL ON FUNCTION realtime.build_prepared_statement_sql(prepared_statement_name text, entity regclass, columns realtime.wal_column[]) TO dashboard_user;


--
-- Name: FUNCTION "cast"(val text, type_ regtype); Type: ACL; Schema: realtime; Owner: supabase_admin
--

GRANT ALL ON FUNCTION realtime."cast"(val text, type_ regtype) TO postgres;
GRANT ALL ON FUNCTION realtime."cast"(val text, type_ regtype) TO dashboard_user;


--
-- Name: FUNCTION check_equality_op(op realtime.equality_op, type_ regtype, val_1 text, val_2 text); Type: ACL; Schema: realtime; Owner: supabase_admin
--

GRANT ALL ON FUNCTION realtime.check_equality_op(op realtime.equality_op, type_ regtype, val_1 text, val_2 text) TO postgres;
GRANT ALL ON FUNCTION realtime.check_equality_op(op realtime.equality_op, type_ regtype, val_1 text, val_2 text) TO dashboard_user;


--
-- Name: FUNCTION is_visible_through_filters(columns realtime.wal_column[], filters realtime.user_defined_filter[]); Type: ACL; Schema: realtime; Owner: supabase_admin
--

GRANT ALL ON FUNCTION realtime.is_visible_through_filters(columns realtime.wal_column[], filters realtime.user_defined_filter[]) TO postgres;
GRANT ALL ON FUNCTION realtime.is_visible_through_filters(columns realtime.wal_column[], filters realtime.user_defined_filter[]) TO dashboard_user;


--
-- Name: FUNCTION quote_wal2json(entity regclass); Type: ACL; Schema: realtime; Owner: supabase_admin
--

GRANT ALL ON FUNCTION realtime.quote_wal2json(entity regclass) TO postgres;
GRANT ALL ON FUNCTION realtime.quote_wal2json(entity regclass) TO dashboard_user;


--
-- Name: FUNCTION subscription_check_filters(); Type: ACL; Schema: realtime; Owner: supabase_admin
--

GRANT ALL ON FUNCTION realtime.subscription_check_filters() TO postgres;
GRANT ALL ON FUNCTION realtime.subscription_check_filters() TO dashboard_user;


--
-- Name: FUNCTION to_regrole(role_name text); Type: ACL; Schema: realtime; Owner: supabase_admin
--

GRANT ALL ON FUNCTION realtime.to_regrole(role_name text) TO postgres;
GRANT ALL ON FUNCTION realtime.to_regrole(role_name text) TO dashboard_user;


```

```
--
-- Name: FUNCTION extension(name text); Type: ACL; Schema: storage; Owner: supabase_storage_admin
--

GRANT ALL ON FUNCTION storage.extension(name text) TO anon;
GRANT ALL ON FUNCTION storage.extension(name text) TO authenticated;
GRANT ALL ON FUNCTION storage.extension(name text) TO service_role;
GRANT ALL ON FUNCTION storage.extension(name text) TO dashboard_user;
GRANT ALL ON FUNCTION storage.extension(name text) TO postgres;


--
-- Name: FUNCTION filename(name text); Type: ACL; Schema: storage; Owner: supabase_storage_admin
--

GRANT ALL ON FUNCTION storage.filename(name text) TO anon;
GRANT ALL ON FUNCTION storage.filename(name text) TO authenticated;
GRANT ALL ON FUNCTION storage.filename(name text) TO service_role;
GRANT ALL ON FUNCTION storage.filename(name text) TO dashboard_user;
GRANT ALL ON FUNCTION storage.filename(name text) TO postgres;


--
-- Name: FUNCTION foldername(name text); Type: ACL; Schema: storage; Owner: supabase_storage_admin
--

GRANT ALL ON FUNCTION storage.foldername(name text) TO anon;
GRANT ALL ON FUNCTION storage.foldername(name text) TO authenticated;
GRANT ALL ON FUNCTION storage.foldername(name text) TO service_role;
GRANT ALL ON FUNCTION storage.foldername(name text) TO dashboard_user;
GRANT ALL ON FUNCTION storage.foldername(name text) TO postgres;


--
-- Name: TABLE audit_log_entries; Type: ACL; Schema: auth; Owner: supabase_auth_admin
--

GRANT ALL ON TABLE auth.audit_log_entries TO dashboard_user;
GRANT ALL ON TABLE auth.audit_log_entries TO postgres;


--
-- Name: TABLE identities; Type: ACL; Schema: auth; Owner: supabase_auth_admin
--

GRANT ALL ON TABLE auth.identities TO postgres;
GRANT ALL ON TABLE auth.identities TO dashboard_user;


--
-- Name: TABLE instances; Type: ACL; Schema: auth; Owner: supabase_auth_admin
--

GRANT ALL ON TABLE auth.instances TO dashboard_user;
GRANT ALL ON TABLE auth.instances TO postgres;


--
-- Name: TABLE refresh_tokens; Type: ACL; Schema: auth; Owner: supabase_auth_admin
--

```

```
3417  GRANT ALL ON TABLE auth.refresh_tokens TO dashboard_user;
3418  GRANT ALL ON TABLE auth.refresh_tokens TO postgres;
3419
3420
3421  --
3422  -- Name: SEQUENCE refresh_tokens_id_seq; Type: ACL; Schema: auth; Owner:
      supabase_auth_admin
3423  --
3424
3425  GRANT ALL ON SEQUENCE auth.refresh_tokens_id_seq TO dashboard_user;
3426  GRANT ALL ON SEQUENCE auth.refresh_tokens_id_seq TO postgres;
3427
3428
3429  --
3430  -- Name: TABLE schema_migrations; Type: ACL; Schema: auth; Owner: supabase_auth_admin
3431  --
3432
3433  GRANT ALL ON TABLE auth.schema_migrations TO dashboard_user;
3434  GRANT ALL ON TABLE auth.schema_migrations TO postgres;
3435
3436
3437  --
3438  -- Name: TABLE users; Type: ACL; Schema: auth; Owner: supabase_auth_admin
3439  --
3440
3441  GRANT ALL ON TABLE auth.users TO dashboard_user;
3442  GRANT ALL ON TABLE auth.users TO postgres;
3443
3444
3445  --
3446  -- Name: TABLE pg_stat_statements; Type: ACL; Schema: extensions; Owner: postgres
3447  --
3448
3449  GRANT ALL ON TABLE extensions.pg_stat_statements TO dashboard_user;
3450
3451
3452  --
3453  -- Name: TABLE pg_stat_statements_info; Type: ACL; Schema: extensions; Owner: postgres
3454  --
3455
3456  GRANT ALL ON TABLE extensions.pg_stat_statements_info TO dashboard_user;
3457
3458
3459  --
3460  -- Name: TABLE "Decisions"; Type: ACL; Schema: public; Owner: supabase_admin
3461  --
3462
3463  GRANT ALL ON TABLE public."Decisions" TO postgres;
3464  GRANT ALL ON TABLE public."Decisions" TO anon;
3465  GRANT ALL ON TABLE public."Decisions" TO authenticated;
3466  GRANT ALL ON TABLE public."Decisions" TO service_role;
3467
3468
3469  --
3470  -- Name: SEQUENCE "Decision_id_seq"; Type: ACL; Schema: public; Owner: supabase_admin
3471  --
3472
3473  GRANT ALL ON SEQUENCE public."Decision_id_seq" TO postgres;
3474  GRANT ALL ON SEQUENCE public."Decision_id_seq" TO anon;
3475  GRANT ALL ON SEQUENCE public."Decision_id_seq" TO authenticated;
3476  GRANT ALL ON SEQUENCE public."Decision_id_seq" TO service_role;
3477
3478
3479  --
```

```
-- Name: TABLE "Donations"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON TABLE public."Donations" TO postgres;
GRANT ALL ON TABLE public."Donations" TO anon;
GRANT ALL ON TABLE public."Donations" TO authenticated;
GRANT ALL ON TABLE public."Donations" TO service_role;


--
-- Name: SEQUENCE "Donation_id_seq"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON SEQUENCE public."Donation_id_seq" TO postgres;
GRANT ALL ON SEQUENCE public."Donation_id_seq" TO anon;
GRANT ALL ON SEQUENCE public."Donation_id_seq" TO authenticated;
GRANT ALL ON SEQUENCE public."Donation_id_seq" TO service_role;


--
-- Name: TABLE "Donors"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON TABLE public."Donors" TO postgres;
GRANT ALL ON TABLE public."Donors" TO anon;
GRANT ALL ON TABLE public."Donors" TO authenticated;
GRANT ALL ON TABLE public."Donors" TO service_role;


--
-- Name: SEQUENCE "Donor_id_seq"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON SEQUENCE public."Donor_id_seq" TO postgres;
GRANT ALL ON SEQUENCE public."Donor_id_seq" TO anon;
GRANT ALL ON SEQUENCE public."Donor_id_seq" TO authenticated;
GRANT ALL ON SEQUENCE public."Donor_id_seq" TO service_role;


--
-- Name: TABLE "Governors"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON TABLE public."Governors" TO postgres;
GRANT ALL ON TABLE public."Governors" TO anon;
GRANT ALL ON TABLE public."Governors" TO authenticated;
GRANT ALL ON TABLE public."Governors" TO service_role;


--
-- Name: SEQUENCE "Governors_id_seq"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON SEQUENCE public."Governors_id_seq" TO postgres;
GRANT ALL ON SEQUENCE public."Governors_id_seq" TO anon;
GRANT ALL ON SEQUENCE public."Governors_id_seq" TO authenticated;
GRANT ALL ON SEQUENCE public."Governors_id_seq" TO service_role;


--
-- Name: TABLE "Organizations"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON TABLE public."Organizations" TO postgres;
```

```
GRANT ALL ON TABLE public."Organizations" TO anon;
GRANT ALL ON TABLE public."Organizations" TO authenticated;
GRANT ALL ON TABLE public."Organizations" TO service_role;


--
-- Name: SEQUENCE "Organizations_id_seq"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON SEQUENCE public."Organizations_id_seq" TO postgres;
GRANT ALL ON SEQUENCE public."Organizations_id_seq" TO anon;
GRANT ALL ON SEQUENCE public."Organizations_id_seq" TO authenticated;
GRANT ALL ON SEQUENCE public."Organizations_id_seq" TO service_role;


--
-- Name: TABLE "Projects"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON TABLE public."Projects" TO postgres;
GRANT ALL ON TABLE public."Projects" TO anon;
GRANT ALL ON TABLE public."Projects" TO authenticated;
GRANT ALL ON TABLE public."Projects" TO service_role;


--
-- Name: SEQUENCE "Projects_id_seq"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON SEQUENCE public."Projects_id_seq" TO postgres;
GRANT ALL ON SEQUENCE public."Projects_id_seq" TO anon;
GRANT ALL ON SEQUENCE public."Projects_id_seq" TO authenticated;
GRANT ALL ON SEQUENCE public."Projects_id_seq" TO service_role;


--
-- Name: TABLE "SingleVotes"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON TABLE public."SingleVotes" TO postgres;
GRANT ALL ON TABLE public."SingleVotes" TO anon;
GRANT ALL ON TABLE public."SingleVotes" TO authenticated;
GRANT ALL ON TABLE public."SingleVotes" TO service_role;


--
-- Name: SEQUENCE "SingleVote_id_seq"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON SEQUENCE public."SingleVote_id_seq" TO postgres;
GRANT ALL ON SEQUENCE public."SingleVote_id_seq" TO anon;
GRANT ALL ON SEQUENCE public."SingleVote_id_seq" TO authenticated;
GRANT ALL ON SEQUENCE public."SingleVote_id_seq" TO service_role;


--
-- Name: TABLE "Solutions"; Type: ACL; Schema: public; Owner: supabase_admin
--

GRANT ALL ON TABLE public."Solutions" TO postgres;
GRANT ALL ON TABLE public."Solutions" TO anon;
GRANT ALL ON TABLE public."Solutions" TO authenticated;
GRANT ALL ON TABLE public."Solutions" TO service_role;
```

```
3607
3608
3609    --
3610    -- Name: SEQUENCE "Solutions_id_seq"; Type: ACL; Schema: public; Owner: supabase_admin
3611    --
3612
3613    GRANT ALL ON SEQUENCE public."Solutions_id_seq" TO postgres;
3614    GRANT ALL ON SEQUENCE public."Solutions_id_seq" TO anon;
3615    GRANT ALL ON SEQUENCE public."Solutions_id_seq" TO authenticated;
3616    GRANT ALL ON SEQUENCE public."Solutions_id_seq" TO service_role;
3617
3618
3619    --
3620    -- Name: TABLE "SystemConstants"; Type: ACL; Schema: public; Owner: supabase_admin
3621    --
3622
3623    GRANT ALL ON TABLE public."SystemConstants" TO postgres;
3624    GRANT ALL ON TABLE public."SystemConstants" TO anon;
3625    GRANT ALL ON TABLE public."SystemConstants" TO authenticated;
3626    GRANT ALL ON TABLE public."SystemConstants" TO service_role;
3627
3628
3629    --
3630    -- Name: SEQUENCE "SystemConstants_id_seq"; Type: ACL; Schema: public; Owner:
        supabase_admin
3631    --
3632
3633    GRANT ALL ON SEQUENCE public."SystemConstants_id_seq" TO postgres;
3634    GRANT ALL ON SEQUENCE public."SystemConstants_id_seq" TO anon;
3635    GRANT ALL ON SEQUENCE public."SystemConstants_id_seq" TO authenticated;
3636    GRANT ALL ON SEQUENCE public."SystemConstants_id_seq" TO service_role;
3637
3638
3639    --
3640    -- Name: TABLE schema_migrations; Type: ACL; Schema: realtime; Owner: supabase_admin
3641    --
3642
3643    GRANT ALL ON TABLE realtime.schema_migrations TO postgres;
3644    GRANT ALL ON TABLE realtime.schema_migrations TO dashboard_user;
3645
3646
3647    --
3648    -- Name: TABLE subscription; Type: ACL; Schema: realtime; Owner: supabase_admin
3649    --
3650
3651    GRANT ALL ON TABLE realtime.subscription TO postgres;
3652    GRANT ALL ON TABLE realtime.subscription TO dashboard_user;
3653
3654
3655    --
3656    -- Name: SEQUENCE subscription_id_seq; Type: ACL; Schema: realtime; Owner: supabase_admin
3657    --
3658
3659    GRANT ALL ON SEQUENCE realtime.subscription_id_seq TO postgres;
3660    GRANT ALL ON SEQUENCE realtime.subscription_id_seq TO dashboard_user;
3661
3662
3663    --
3664    -- Name: TABLE buckets; Type: ACL; Schema: storage; Owner: supabase_storage_admin
3665    --
3666
3667    GRANT ALL ON TABLE storage.buckets TO anon;
3668    GRANT ALL ON TABLE storage.buckets TO authenticated;
3669    GRANT ALL ON TABLE storage.buckets TO service_role;
```

```
3670  GRANT ALL ON TABLE storage.buckets TO postgres;
3671
3672
3673  --
3674  -- Name: TABLE migrations; Type: ACL; Schema: storage; Owner: supabase_storage_admin
3675  --
3676
3677  GRANT ALL ON TABLE storage.migrations TO anon;
3678  GRANT ALL ON TABLE storage.migrations TO authenticated;
3679  GRANT ALL ON TABLE storage.migrations TO service_role;
3680  GRANT ALL ON TABLE storage.migrations TO postgres;
3681
3682
3683  --
3684  -- Name: TABLE objects; Type: ACL; Schema: storage; Owner: supabase_storage_admin
3685  --
3686
3687  GRANT ALL ON TABLE storage.objects TO anon;
3688  GRANT ALL ON TABLE storage.objects TO authenticated;
3689  GRANT ALL ON TABLE storage.objects TO service_role;
3690  GRANT ALL ON TABLE storage.objects TO postgres;
3691
3692
3693  --
3694  -- Name: DEFAULT PRIVILEGES FOR SEQUENCES; Type: DEFAULT ACL; Schema: auth; Owner:
     supabase_auth_admin
3695  --
3696
3697  ALTER DEFAULT PRIVILEGES FOR ROLE supabase_auth_admin IN SCHEMA auth GRANT ALL ON
     SEQUENCES  TO postgres;
3698  ALTER DEFAULT PRIVILEGES FOR ROLE supabase_auth_admin IN SCHEMA auth GRANT ALL ON
     SEQUENCES  TO dashboard_user;
3699
3700
3701  --
3702  -- Name: DEFAULT PRIVILEGES FOR FUNCTIONS; Type: DEFAULT ACL; Schema: auth; Owner:
     supabase_auth_admin
3703  --
3704
3705  ALTER DEFAULT PRIVILEGES FOR ROLE supabase_auth_admin IN SCHEMA auth GRANT ALL ON
     FUNCTIONS  TO postgres;
3706  ALTER DEFAULT PRIVILEGES FOR ROLE supabase_auth_admin IN SCHEMA auth GRANT ALL ON
     FUNCTIONS  TO dashboard_user;
3707
3708
3709  --
3710  -- Name: DEFAULT PRIVILEGES FOR TABLES; Type: DEFAULT ACL; Schema: auth; Owner:
     supabase_auth_admin
3711  --
3712
3713  ALTER DEFAULT PRIVILEGES FOR ROLE supabase_auth_admin IN SCHEMA auth GRANT ALL ON TABLES
       TO postgres;
3714  ALTER DEFAULT PRIVILEGES FOR ROLE supabase_auth_admin IN SCHEMA auth GRANT ALL ON TABLES
       TO dashboard_user;
3715
3716
3717  --
3718  -- Name: DEFAULT PRIVILEGES FOR SEQUENCES; Type: DEFAULT ACL; Schema: graphql; Owner:
     supabase_admin
3719  --
3720
3721  ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON
     SEQUENCES  TO postgres;
3722  ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON
```

```
SEQUENCES  TO anon;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON SEQUENCES  TO authenticated;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON SEQUENCES  TO service_role;


--
-- Name: DEFAULT PRIVILEGES FOR FUNCTIONS; Type: DEFAULT ACL; Schema: graphql; Owner: supabase_admin
--

ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON FUNCTIONS  TO postgres;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON FUNCTIONS  TO anon;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON FUNCTIONS  TO authenticated;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON FUNCTIONS  TO service_role;


--
-- Name: DEFAULT PRIVILEGES FOR TABLES; Type: DEFAULT ACL; Schema: graphql; Owner: supabase_admin
--

ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON TABLES TO postgres;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON TABLES TO anon;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON TABLES TO authenticated;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql GRANT ALL ON TABLES TO service_role;


--
-- Name: DEFAULT PRIVILEGES FOR SEQUENCES; Type: DEFAULT ACL; Schema: graphql_public; Owner: supabase_admin
--

ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON SEQUENCES  TO postgres;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON SEQUENCES  TO anon;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON SEQUENCES  TO authenticated;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON SEQUENCES  TO service_role;


--
-- Name: DEFAULT PRIVILEGES FOR FUNCTIONS; Type: DEFAULT ACL; Schema: graphql_public; Owner: supabase_admin
--

ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON FUNCTIONS  TO postgres;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON FUNCTIONS  TO anon;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON FUNCTIONS  TO authenticated;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON
```

```
    FUNCTIONS  TO service_role;
3765
3766
3767    --
3768    -- Name: DEFAULT PRIVILEGES FOR TABLES; Type: DEFAULT ACL; Schema: graphql_public;
    Owner: supabase_admin
3769    --
3770
3771    ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON
    TABLES  TO postgres;
3772    ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON
    TABLES  TO anon;
3773    ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON
    TABLES  TO authenticated;
3774    ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA graphql_public GRANT ALL ON
    TABLES  TO service_role;
3775
3776
3777    --
3778    -- Name: DEFAULT PRIVILEGES FOR SEQUENCES; Type: DEFAULT ACL; Schema: public; Owner:
    postgres
3779    --
3780
3781    ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON SEQUENCES  TO
    postgres;
3782    ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON SEQUENCES  TO
    anon;
3783    ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON SEQUENCES  TO
    authenticated;
3784    ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON SEQUENCES  TO
    service_role;
3785
3786
3787    --
3788    -- Name: DEFAULT PRIVILEGES FOR SEQUENCES; Type: DEFAULT ACL; Schema: public; Owner:
    supabase_admin
3789    --
3790
3791    ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON SEQUENCES
      TO postgres;
3792    ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON SEQUENCES
      TO anon;
3793    ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON SEQUENCES
      TO authenticated;
3794    ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON SEQUENCES
      TO service_role;
3795
3796
3797    --
3798    -- Name: DEFAULT PRIVILEGES FOR FUNCTIONS; Type: DEFAULT ACL; Schema: public; Owner:
    postgres
3799    --
3800
3801    ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON FUNCTIONS  TO
    postgres;
3802    ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON FUNCTIONS  TO
    anon;
3803    ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON FUNCTIONS  TO
    authenticated;
3804    ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON FUNCTIONS  TO
    service_role;
3805
3806
3807    --
```

```
-- Name: DEFAULT PRIVILEGES FOR FUNCTIONS; Type: DEFAULT ACL; Schema: public; Owner: supabase_admin
--

ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON FUNCTIONS  TO postgres;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON FUNCTIONS  TO anon;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON FUNCTIONS  TO authenticated;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON FUNCTIONS  TO service_role;


--
-- Name: DEFAULT PRIVILEGES FOR TABLES; Type: DEFAULT ACL; Schema: public; Owner: postgres
--

ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON TABLES  TO postgres;
ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON TABLES  TO anon;
ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON TABLES  TO authenticated;
ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA public GRANT ALL ON TABLES  TO service_role;


--
-- Name: DEFAULT PRIVILEGES FOR TABLES; Type: DEFAULT ACL; Schema: public; Owner: supabase_admin
--

ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON TABLES  TO postgres;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON TABLES  TO anon;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON TABLES  TO authenticated;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA public GRANT ALL ON TABLES  TO service_role;


--
-- Name: DEFAULT PRIVILEGES FOR SEQUENCES; Type: DEFAULT ACL; Schema: realtime; Owner: supabase_admin
--

ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA realtime GRANT ALL ON SEQUENCES  TO postgres;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA realtime GRANT ALL ON SEQUENCES  TO dashboard_user;


--
-- Name: DEFAULT PRIVILEGES FOR FUNCTIONS; Type: DEFAULT ACL; Schema: realtime; Owner: supabase_admin
--

ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA realtime GRANT ALL ON FUNCTIONS  TO postgres;
ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA realtime GRANT ALL ON FUNCTIONS  TO dashboard_user;

```

```
3852
3853  --
3854  -- Name: DEFAULT PRIVILEGES FOR TABLES; Type: DEFAULT ACL; Schema: realtime; Owner:
      supabase_admin
3855  --
3856
3857  ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA realtime GRANT ALL ON TABLES
        TO postgres;
3858  ALTER DEFAULT PRIVILEGES FOR ROLE supabase_admin IN SCHEMA realtime GRANT ALL ON TABLES
        TO dashboard_user;
3859
3860
3861  --
3862  -- Name: DEFAULT PRIVILEGES FOR SEQUENCES; Type: DEFAULT ACL; Schema: storage; Owner:
      postgres
3863  --
3864
3865  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON SEQUENCES  TO
      postgres;
3866  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON SEQUENCES  TO
      anon;
3867  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON SEQUENCES  TO
      authenticated;
3868  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON SEQUENCES  TO
      service_role;
3869
3870
3871  --
3872  -- Name: DEFAULT PRIVILEGES FOR FUNCTIONS; Type: DEFAULT ACL; Schema: storage; Owner:
      postgres
3873  --
3874
3875  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON FUNCTIONS  TO
      postgres;
3876  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON FUNCTIONS  TO
      anon;
3877  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON FUNCTIONS  TO
      authenticated;
3878  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON FUNCTIONS  TO
      service_role;
3879
3880
3881  --
3882  -- Name: DEFAULT PRIVILEGES FOR TABLES; Type: DEFAULT ACL; Schema: storage; Owner:
      postgres
3883  --
3884
3885  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON TABLES  TO
      postgres;
3886  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON TABLES  TO
      anon;
3887  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON TABLES  TO
      authenticated;
3888  ALTER DEFAULT PRIVILEGES FOR ROLE postgres IN SCHEMA storage GRANT ALL ON TABLES  TO
      service_role;
3889
3890
3891  --
3892  -- Name: issue_graphql_placeholder; Type: EVENT TRIGGER; Schema: -; Owner: supabase_admin
3893  --
3894
3895  CREATE EVENT TRIGGER issue_graphql_placeholder ON sql_drop
3896           WHEN TAG IN ('DROP EXTENSION')
3897     EXECUTE FUNCTION extensions.set_graphql_placeholder();
```

```
3898
3899
3900  ALTER EVENT TRIGGER issue_graphql_placeholder OWNER TO supabase_admin;
3901
3902  --
3903  -- Name: issue_pg_cron_access; Type: EVENT TRIGGER; Schema: -; Owner: postgres
3904  --
3905
3906  CREATE EVENT TRIGGER issue_pg_cron_access ON ddl_command_end
3907           WHEN TAG IN ('CREATE SCHEMA')
3908     EXECUTE FUNCTION extensions.grant_pg_cron_access();
3909
3910
3911  ALTER EVENT TRIGGER issue_pg_cron_access OWNER TO postgres;
3912
3913  --
3914  -- Name: issue_pg_graphql_access; Type: EVENT TRIGGER; Schema: -; Owner: supabase_admin
3915  --
3916
3917  CREATE EVENT TRIGGER issue_pg_graphql_access ON ddl_command_end
3918           WHEN TAG IN ('CREATE FUNCTION')
3919     EXECUTE FUNCTION extensions.grant_pg_graphql_access();
3920
3921
3922  ALTER EVENT TRIGGER issue_pg_graphql_access OWNER TO supabase_admin;
3923
3924  --
3925  -- Name: issue_pg_net_access; Type: EVENT TRIGGER; Schema: -; Owner: postgres
3926  --
3927
3928  CREATE EVENT TRIGGER issue_pg_net_access ON ddl_command_end
3929           WHEN TAG IN ('CREATE EXTENSION')
3930     EXECUTE FUNCTION extensions.grant_pg_net_access();
3931
3932
3933  ALTER EVENT TRIGGER issue_pg_net_access OWNER TO postgres;
3934
3935  --
3936  -- Name: pgrst_ddl_watch; Type: EVENT TRIGGER; Schema: -; Owner: supabase_admin
3937  --
3938
3939  CREATE EVENT TRIGGER pgrst_ddl_watch ON ddl_command_end
3940     EXECUTE FUNCTION extensions.pgrst_ddl_watch();
3941
3942
3943  ALTER EVENT TRIGGER pgrst_ddl_watch OWNER TO supabase_admin;
3944
3945  --
3946  -- Name: pgrst_drop_watch; Type: EVENT TRIGGER; Schema: -; Owner: supabase_admin
3947  --
3948
3949  CREATE EVENT TRIGGER pgrst_drop_watch ON sql_drop
3950     EXECUTE FUNCTION extensions.pgrst_drop_watch();
3951
3952
3953  ALTER EVENT TRIGGER pgrst_drop_watch OWNER TO supabase_admin;
3954
3955  --
3956  -- PostgreSQL database dump complete
3957  --
3958
3959
```