Physical Data Diagram

Finnova Schema

Generated: 3/9/2019 4:03:23 PM

Table of Contents

---

1. Diagram Information

1.1. Basic Information

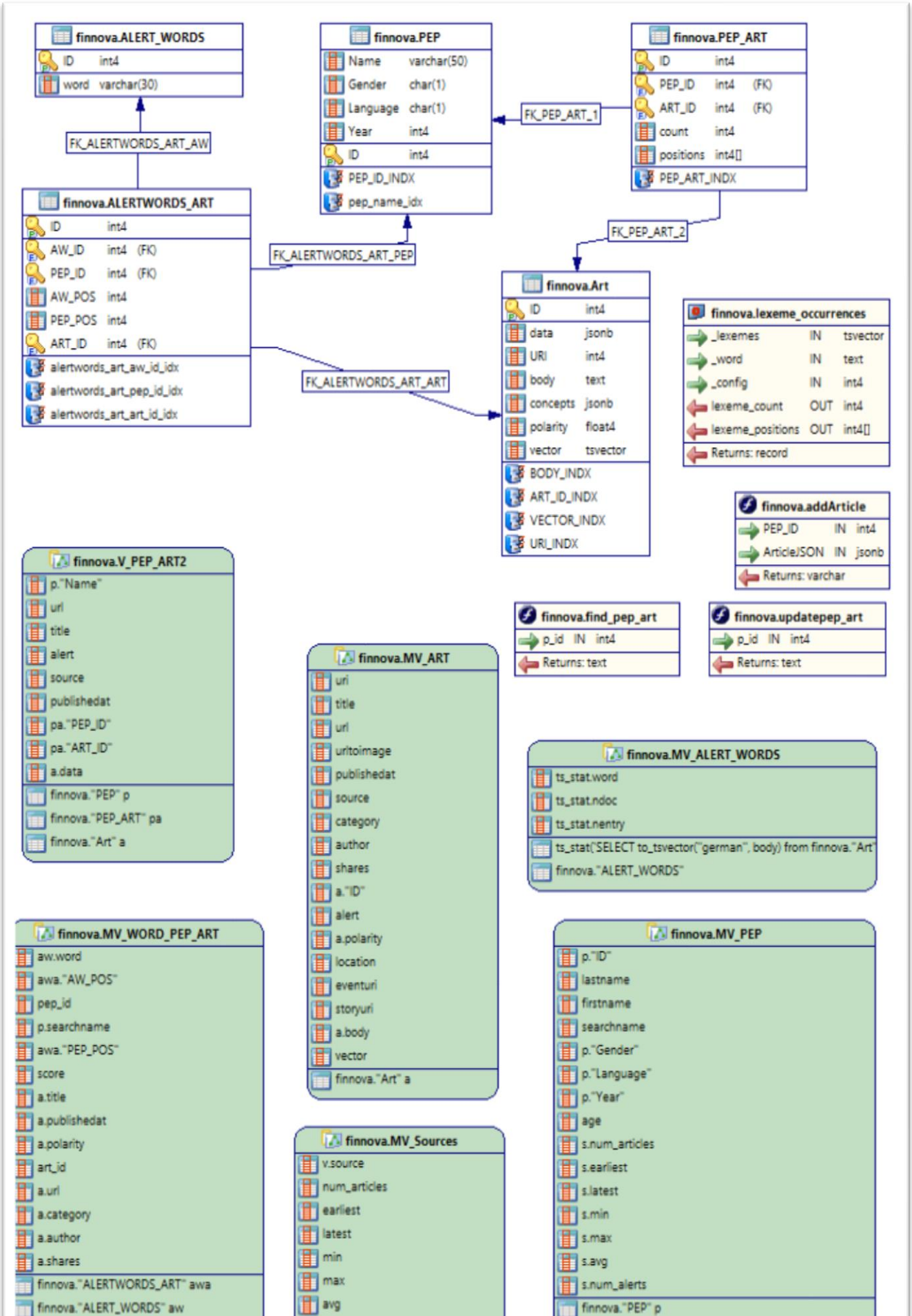
Project	Finnova Adverse Media Search
Diagram	Finnova Schema
Company	Sequoia Intelligence
Author	Bil Worthington
Version	1.0
Created	3/9/2019 3:33:52 PM
Last Modified	3/9/2019 3:33:52 PM

1.2. Diagram Description

```
The diagram below is for the Finnova Adverse Media Search
 Big Data Analytics project.
It shows the entity relationships between the tables.
Stored functions and Materialized Views.
```

1.3. Diagram Annotation

1.4. ER Diagram


finnova.ALERT_WORDS
ID int4
word varchar(30)
FK_ALERTWORDS_ART_AW
finnova.ALERTWORDS_ART
ID int4
AW_ID int4 (FK)
PEP_ID int4 (FK)
AW_POS int4
PEP_POS int4
ART_ID int4 (FK)
alertwords_art_aw_id_idx
alertwords_art_pep_id_idx
alertwords_art_art_id_idx
FK_ALERTWORDS_ART_PEP
FK_ALERTWORDS_ART_ART
finnova.PEP
Name varchar(50)
Gender char(1)
Language char(1)
Year int4
ID int4
PEP_ID_INDX
pep_name_idx
FK_PEP_ART_1
finnova.PEP_ART
ID int4
PEP_ID int4 (FK)
ART_ID int4 (FK)
count int4
positions int4[]
PEP_ART_INDX
FK_PEP_ART_2
finnova.Art
ID int4
data jsonb
URI int4
body text
concepts jsonb
polarity float4
vector tsvector
BODY_INDX
ART_ID_INDX
VECTOR_INDX
URI_INDX
finnova.lexeme_occurrences
_lexemes IN tsvector
_word IN text
_config IN int4
lexeme_count OUT int4
lexeme_positions OUT int4[]
Returns: record
finnova.addArticle
PEP_ID IN int4
ArticleJSON IN jsonb
Returns: varchar
finnova.find_pep_art
p_id IN int4
Returns: text
finnova.updatepep_art
p_id IN int4
Returns: text
finnova.V_PEP_ART2
p."Name"
url
title
alert
source
publishedat
pa."PEP_ID"
pa."ART_ID"
a.data
finnova."PEP" p
finnova."PEP_ART" pa
finnova."Art" a
finnova.MV_ART
uri
title
url
uritoimage
publishedat
source
category
author
shares
a."ID"
alert
a.polarity
location
eventuri
storyuri
a.body
vector
finnova."Art" a
finnova.MV_ALERT_WORDS
ts_stat.word
ts_stat.ndoc
ts_stat.nentry
ts_stat('SELECT to_tsvector("german", body) from finnova."Art"
finnova."ALERT_WORDS"
finnova.MV_WORD_PEP_ART
aw.word
awa."AW_POS"
pep_id
p.searchname
awa."PEP_POS"
score
a.title
a.publishedat
a.polarity
art_id
a.url
a.category
a.author
a.shares
finnova."ALERTWORDS_ART" awa
finnova."ALERT_WORDS" aw
finnova.MV_Sources
v.source
num_articles
earliest
latest
min
max
avg
finnova.MV_PEP
p."ID"
lastname
firstname
searchname
p."Gender"
p."Language"
p."Year"
age
s.num_articles
s.earliest
s.latest
s.min
s.max
s.avg
s.num_alerts
finnova."PEP" p

2. Database

Description	This data base is for storing data for the Finnova Adverse Media Search Big Data Analytics project. It runs on Amazon RDS system, t2.micro instance was found to be adquite to run this DB. This DB was designed to be a Online Analytical Processing (OLAP) system. After data is written, it is not expected to be changed, just read. The data base is heavily indexed for fast text searches. Materialized Views are used to help increase the performance.
Annotation	
Name	Finnova
Character Set	
Template	
Tablespace	
Comment	Database for the Big Data Analytics Finnova Adverse Media Search project. This data base is for storing data for the Finnova Adverse Media Search Big Data Analytics project. It runs on Amazon RDS system, t2.micro instance was found to be adquite to run this DB. This DB was designed to be a Online Analytical Processing (OLAP) system. After data is written, it is not expected to be changed, just read. The data base is heavily indexed for fast text searches. Materialized Views are used to help increase the performance.
Create SQL	`CREATE DATABASE "Finnova"` `  OWNER = "FreiburgBill";`

3 Domains

4. Tables

4.1 ALERTWORDS_ART

Description	
Annotation	
Comment	This table joins the Alert_Words table with the PEP table. It identifies the position of an alert word in an article, the PEP that is closes to that word, and the position in the article of that PEP.
Owner	FreiburgBill

Temporary	With OIDs	Fill Factor
-	False	

Columns	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
ID	int4	✔	✔	✔			Primiary Key ID
AW_ID	int4	-	-	-			FK to Alert_Words ID
PEP_ID	int4	-	-	-			FK to PEP ID
AW_POS	int4	-	-	-			Position of the alert word in the article by word count.
PEP_POS	int4	-	-	-			Position of the PEP's last name in the article that is closes to the alert word.
ART_ID	int4	-	-	-			FK on Art table ID field.

Indexes	Unique	Columns	Method	Comment
alertwords_art_aw_id_idx	-	AW_ID	BTREE	
alertwords_art_pep_id_idx	-	PEP_ID	BTREE	
alertwords_art_art_id_idx	-	ART_ID	BTREE	

Constraints	Kind	Expression	Columns	Comment
ALERTWORDS_ART_pkey	PRIMARY KEY		ID	

Rules	Kind	Instead	Expression	Body	Comment

Triggers	Time	Procedure	Events	For Each Row	Comment

Create SQL

```
CREATE TABLE "finnova"."ALERTWORDS_ART" (
        "ID" SERIAL NOT NULL,
        "AW_ID" int4,
        "PEP_ID" int4,
        "AW_POS" int4,
        "PEP_POS" int4,
        "ART_ID" int4,
        CONSTRAINT "ALERTWORDS_ART_pkey" PRIMARY KEY("ID"),
        CONSTRAINT "FK_ALERTWORDS_ART_AW" FOREIGN KEY ("AW_ID")
                REFERENCES "finnova"."ALERT_WORDS"("ID")
        MATCH SIMPLE
        ON DELETE NO ACTION
```

```
        ON UPDATE NO ACTION
        NOT DEFERRABLE,
        CONSTRAINT "FK_ALERTWORDS_ART_PEP" FOREIGN KEY ("PEP_ID")
                REFERENCES "finnova"."PEP"("ID")
        MATCH SIMPLE
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
        NOT DEFERRABLE,
        CONSTRAINT "FK_ALERTWORDS_ART_ART" FOREIGN KEY ("ART_ID")
                REFERENCES "finnova"."Art"("ID")
        MATCH SIMPLE
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
        NOT DEFERRABLE
)
WITH (
        OIDS = False
);

CREATE INDEX "alertwords_art_aw_id_idx" ON "finnova"."ALERTWORDS_ART"
USING BTREE (
        "AW_ID"
);


CREATE INDEX "alertwords_art_pep_id_idx" ON "finnova"."ALERTWORDS_ART"
USING BTREE (
        "PEP_ID"
);


CREATE INDEX "alertwords_art_art_id_idx" ON "finnova"."ALERTWORDS_ART"
USING BTREE (
        "ART_ID"
);


ALTER TABLE "finnova"."ALERTWORDS_ART" OWNER TO "FreiburgBill";

COMMENT ON TABLE "finnova"."ALERTWORDS_ART" IS 'This table joins the
Alert_Words table with the PEP table.
It identifies the position of an alert word in an article, the PEP that
is closes to that word, and the position in the article of that PEP.';

COMMENT ON COLUMN "finnova"."ALERTWORDS_ART"."ID" IS 'Primiary Key ID';

COMMENT ON COLUMN "finnova"."ALERTWORDS_ART"."AW_ID" IS 'FK to
Alert_Words ID';

COMMENT ON COLUMN "finnova"."ALERTWORDS_ART"."PEP_ID" IS 'FK to PEP ID';

COMMENT ON COLUMN "finnova"."ALERTWORDS_ART"."AW_POS" IS 'Position of
the alert word in the article by word count.';

COMMENT ON COLUMN "finnova"."ALERTWORDS_ART"."PEP_POS" IS 'Position of
the PEP''s last name in the article that is closes to the alert word.';

COMMENT ON COLUMN "finnova"."ALERTWORDS_ART"."ART_ID" IS 'FK on Art
table ID field.';
```

4.2 ALERT_WORDS

Description	
Annotation	
Comment	
Owner	FreiburgBill

Temporary	With OIDs	Fill Factor
-	False	

Columns	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
ID	int4	✔	✔	✔			
word	varchar(30)	-	✔	-			

Indexes	Unique	Columns	Method	Comment

Constraints	Kind	Expression	Columns	Comment
ALERT_WORDS_pkey	PRIMARY KEY		ID	

Rules	Kind	Instead	Expression	Body	Comment

Triggers	Time	Procedure	Events	For Each Row	Comment

Create SQL

```
CREATE TABLE "finnova"."ALERT_WORDS" (
        "ID" SERIAL NOT NULL,
        "word" varchar(30) NOT NULL,
        CONSTRAINT "ALERT_WORDS_pkey" PRIMARY KEY("ID")
)
WITH (
        OIDS = False
);

ALTER TABLE "finnova"."ALERT_WORDS" OWNER TO "FreiburgBill";
```

4.3 Art

Description	
Annotation	
Comment	Table for EventRegistry Articles
Owner	FreiburgBill

Temporary	With OIDs	Fill Factor
-	False	

Columns	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
ID	int4	✔	✔	✔			Primary Key
data	jsonb	-	-	-			Contains returned JSON object from EventRegisty
URI	int4	-	✔	-			Article unique URI
body	text	-	-	-			The article text removed from the EventRegistry JSON
concepts	jsonb	-	-	-			JSON concepts object removed from EventRegistry JSON
polarity	float4	-	-	-			Sentiment Polarity

vector	tsvector	-	-	-		TSVECTOR of the article body.

Indexes	Unique	Columns	Method	Comment
BODY_INDX	-	body	GIST	
ART_ID_INDX	✔	ID	BTREE	
VECTOR_INDX	-	vector	GIN	GIN Index on the tsvector column 'Vector2'
URI_INDX	✔	URI	BTREE	Unique URI index. Article URI should only exist once in the table.

Constraints	Kind	Expression	Columns	Comment
Art_pkey	PRIMARY KEY		ID	

Rules	Kind	Instead	Expression	Body	Comment

Triggers	Time	Procedure	Events	For Each Row	Comment

Create SQL

```
CREATE TABLE "finnova"."Art" (
        "ID" SERIAL NOT NULL,
        "data" jsonb,
        "URI" int4 NOT NULL,
        "body" text,
        "concepts" jsonb,
        "polarity" float4,
        "vector" tsvector,
        CONSTRAINT "Art_pkey" PRIMARY KEY("ID")
)
WITH (
        OIDS = False
);

CREATE INDEX "BODY_INDX" ON "finnova"."Art" USING GIST (
        "body"
)
WITH (
        BUFFERING = AUTO
);


CREATE UNIQUE INDEX "ART_ID_INDX" ON "finnova"."Art" USING BTREE (
        "ID"
);


CREATE INDEX "VECTOR_INDX" ON "finnova"."Art" USING GIN (
        "vector"
)
WITH (
        FASTUPDATE = ON
);


CREATE UNIQUE INDEX "URI_INDX" ON "finnova"."Art" USING BTREE (
        "URI"
);


ALTER TABLE "finnova"."Art" OWNER TO "FreiburgBill";

COMMENT ON TABLE "finnova"."Art" IS 'Table for EventRegistry Articles';
```

```
COMMENT ON COLUMN "finnova"."Art"."ID" IS 'Primary Key';

COMMENT ON COLUMN "finnova"."Art"."data" IS 'Contains returned JSON object
from EventRegisty';

COMMENT ON COLUMN "finnova"."Art"."URI" IS 'Article unique URI';

COMMENT ON COLUMN "finnova"."Art"."body" IS 'The article text removed from
the EventRegistry JSON';

COMMENT ON COLUMN "finnova"."Art"."concepts" IS 'JSON concepts object
removed from EventRegistry JSON';

COMMENT ON COLUMN "finnova"."Art"."polarity" IS 'Sentiment Polarity ';

COMMENT ON COLUMN "finnova"."Art"."vector" IS 'TSVECTOR of the article
body.';

COMMENT ON INDEX "finnova"."VECTOR_INDX" IS 'GIN Index on the tsvector
column ''Vector2''';

COMMENT ON INDEX "finnova"."URI_INDX" IS 'Unique URI index.
Article URI should only exist once in the table.';
```

4.4 PEP

Description	
Annotation	
Comment	
Owner	FreiburgBill

Temporary	With OIDs	Fill Factor
-	False	

Columns	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
Name	varchar(50)	-	✔	-			LastName FirstName
Gender	char(1)	-	-	-			M - Male F- Female
Language	char(1)	-	-	-			d - Deutch f - French
Year	int4	-	-	-			Year of birth
ID	int4	✔	✔	✔			Primary Key

Indexes	Unique	Columns	Method	Comment
PEP_ID_INDX	✔	ID	BTREE	
pep_name_idx	✔	Name	BTREE	

Constraints	Kind	Expression	Columns	Comment
PEP_pkey	PRIMARY KEY		ID	

Rules	Kind	Instead	Expression	Body	Comment

Triggers	Time	Procedure	Events	For Each Row	Comment

Create SQL

```
CREATE TABLE "finnova"."PEP" (
        "Name" varchar(50) NOT NULL,
        "Gender" char(1),
```

```
        "Language" char(1),
        "Year" int4,
        "ID" SERIAL NOT NULL,
        CONSTRAINT "PEP_pkey" PRIMARY KEY("ID")
)
WITH (
        OIDS = False
);

CREATE UNIQUE INDEX "PEP_ID_INDX" ON "finnova"."PEP" USING BTREE (
        "ID"
);


CREATE UNIQUE INDEX "pep_name_idx" ON "finnova"."PEP" USING BTREE (
        "Name"
);


ALTER TABLE "finnova"."PEP" OWNER TO "FreiburgBill";

COMMENT ON COLUMN "finnova"."PEP"."Name" IS 'LastName FirstName';

COMMENT ON COLUMN "finnova"."PEP"."Gender" IS 'M - Male
F- Female ';

COMMENT ON COLUMN "finnova"."PEP"."Language" IS 'd - Deutch
f -  French';

COMMENT ON COLUMN "finnova"."PEP"."Year" IS 'Year of birth';

COMMENT ON COLUMN "finnova"."PEP"."ID" IS 'Primary Key';
```

4.5 PEP_ART

Description	
Annotation	
Comment	Table to join a PEP to Articles they are mentioned in.
Owner	FreiburgBill

Temporary	With OIDs	Fill Factor
-	False	

Columns	Data Type	Primary Key	Not Null	AutoInc	Flags	Default Value	Comment
ID	int4	✔	✔	✔			
PEP_ID	int4	-	✔	-			Foreign Key to PEP table ID column.
ART_ID	int4	-	✔	-			Foreign Key to ART table ID column.
count	int4	-	-	-			Number of times this PEP (PEP_ID) appears in this article.
positions	int4[]	-	-	-			Integer array of the positions (by word count) where this PEP's last name appears.

Indexes	Unique	Columns	Method	Comment
PEP_ART_INDX	✔	PEP_ID,ART_ID	BTREE	

Constraints	Kind	Expression	Columns	Comment
PEP_ART_pkey	PRIMARY KEY		ID	

Rules	Kind	Instead	Expression	Body	Comment

Triggers	Time	Procedure	Events	For Each Row	Comment

Create SQL

```
CREATE TABLE "finnova"."PEP_ART" (
        "ID" SERIAL NOT NULL,
        "PEP_ID" int4 NOT NULL,
        "ART_ID" int4 NOT NULL,
        "count" int4,
        "positions" int4[],
        CONSTRAINT "PEP_ART_pkey" PRIMARY KEY("ID"),
        CONSTRAINT "FK_PEP_ART_1" FOREIGN KEY ("PEP_ID")
                REFERENCES "finnova"."PEP"("ID")
        MATCH SIMPLE
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
        NOT DEFERRABLE,
        CONSTRAINT "FK_PEP_ART_2" FOREIGN KEY ("ART_ID")
                REFERENCES "finnova"."Art"("ID")
        MATCH SIMPLE
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
        NOT DEFERRABLE
)
WITH (
        OIDS = False
);

CREATE UNIQUE INDEX "PEP_ART_INDX" ON "finnova"."PEP_ART" USING BTREE (
        "PEP_ID",
        "ART_ID"
);


ALTER TABLE "finnova"."PEP_ART" OWNER TO "FreiburgBill";

COMMENT ON TABLE "finnova"."PEP_ART" IS 'Table to join a PEP to Articles
they are mentioned in.';

COMMENT ON COLUMN "finnova"."PEP_ART"."PEP_ID" IS 'Foreign Key to PEP
table ID column.';

COMMENT ON COLUMN "finnova"."PEP_ART"."ART_ID" IS 'Foreign Key to ART
table ID column.';

COMMENT ON COLUMN "finnova"."PEP_ART"."count" IS 'Number of times this PEP
(PEP_ID) appears in this article.';

COMMENT ON COLUMN "finnova"."PEP_ART"."positions" IS 'Integer array of the
positions (by word count) where this PEP''s last name appears.';
```

5. References

5.1 FK_ALERTWORDS_ART_AW

Description

Annotation

Parent Table	Child Table	Delete Action	Update Action	Link
ALERT_WORDS	ALERTWORDS_ART	NO ACTION	NO ACTION	ID=AW_ID

5.2 FK_ALERTWORDS_ART_PEP

Description

Annotation

Parent Table	Child Table	Delete Action	Update Action	Link
PEP	ALERTWORDS_ART	NO ACTION	NO ACTION	ID=PEP_ID

5.3 FK_ALERTWORDS_ART_ART

Description

Annotation

Parent Table	Child Table	Delete Action	Update Action	Link
Art	ALERTWORDS_ART	NO ACTION	NO ACTION	ID=ART_ID

5.4 FK_PEP_ART_1

Description

Annotation

Parent Table	Child Table	Delete Action	Update Action	Link
PEP	PEP_ART	NO ACTION	NO ACTION	ID=PEP_ID

5.5 FK_PEP_ART_2

Description

Annotation

Parent Table	Child Table	Delete Action	Update Action	Link
Art	PEP_ART	NO ACTION	NO ACTION	ID=ART_ID

6. Stored Procedures

6.1 lexeme_occurrences

Description

Annotation

Comment

```
Parameters in :
_lexemes a tsvector that is to be searched
-word  a word to search the tesvector for
-config  language config, e.g. 'german'
Parameters out:
lexeme_count number found
lexeme_positions integer array of positions where they were found.
```

Owner FreiburgBill

Param Name	Param Type	Data Type
_lexemes	IN	tsvector
_word	IN	text
_config	IN	int4
lexeme_count	OUT	int4
lexeme_positions	OUT	int4[]

Create SQL

```
CREATE OR REPLACE FUNCTION "finnova"."lexeme_occurrences" (IN _lexemes
tsvector,
        IN _word text,
        IN _config int4,
        OUT lexeme_count int4,
        OUT lexeme_positions int4[])
RETURNS record AS
$BODY$
DECLARE
--   _lexemes             tsvector := to_tsvector (_config, _document);
   _searched_lexeme      tsvector :=  strip (to_tsvector (_config,
_word));
   _occurences_pattern   TEXT := _searched_lexeme::TEXT || ':([0-9,]+)';
   _occurences_list      TEXT
      := substring (_lexemes::TEXT, _occurences_pattern);
BEGIN
   SELECT count (a), array_agg (a::INT)
     FROM regexp_split_to_table (_occurences_list, ',') a
    WHERE _searched_lexeme::TEXT != ''           -- preventing false
positives
     INTO lexeme_count, lexeme_positions;

   RETURN;
END
$BODY$
        LANGUAGE plpgsql
        CALLED ON NULL INPUT
        VOLATILE
        EXTERNAL SECURITY INVOKER
        COST 100;

COMMENT ON FUNCTION "finnova"."lexeme_occurrences" (IN _lexemes tsvector,
        IN _word text,
        IN _config int4,
        OUT lexeme_count int4,
        OUT lexeme_positions int4[]) IS 'Parameters in :
```

```
_lexemes a tsvector that is to be searched
-word  a word to search the tesvector for
-config  language config, e.g. ''german''
Parameters out:
lexeme_count number found
lexeme_positions integer array of positions where they were found.';

ALTER FUNCTION "finnova"."lexeme_occurrences" (IN _lexemes tsvector,
        IN _word text,
        IN _config int4,
        OUT lexeme_count int4,
        OUT lexeme_positions int4[]) OWNER TO "FreiburgBill";
```

6.2 addArticle

Description	
Annotation	
Comment	This function is used for adding articles and associating a PEP with them. It first checks if there are any articles already in the database that are at least 80% similar to the new one with the PEP in them. If articles are found that are mroe than 80% similar to the passed in one, the passed in PEP is associated with the most simlar article in the PEP_ART table. If no similar articles are found, the new one will be inserted into the Art table.
Owner	FreiburgBill

Param Name	Param Type	Data Type
PEP_ID	IN	int4
ArticleJSON	IN	jsonb

Create SQL

```
CREATE OR REPLACE FUNCTION "finnova"."addArticle" (IN PEP_ID int4, IN
ArticleJSON jsonb)
RETURNS varchar AS
$BODY$
DECLARE
   retMessg     CHARACTER VARYING;
   errMessg     CHARACTER VARYING;
   art_id       INTEGER;
   pep_art_id   INTEGER;
   uriInt       INTEGER;
   countDups    INTEGER;
   jsonURI      TEXT;
   jsonBody     TEXT;
   data         JSONB;
   concepts     JSONB;
BEGIN
   uriInt = ("ArticleJSON" ->> 'uri'::TEXT);
   jsonBody = ("ArticleJSON" ->> 'body'::TEXT);
   concepts = ("ArticleJSON" ->> 'concepts'::TEXT);
   retMessg = "ArticleJSON" -> 'title';

   -- see if any existing articles with the current PEP in the them are
   -- more than 80% similar to this new one,
   -- if so, consider it a duplicate
   -- get the art_id of the most similar article

        SELECT s."ID"
```

```
        INTO art_id
        FROM (
                SELECT "ID", similarity(body, jsonBody)
                FROM finnova."Art" a
                WHERE a."body" LIKE (SELECT '%' || lastname || '%'
                                                                FROM
finnova."MV_PEP"
                                                                WHERE "ID" =
"PEP_ID")               -- only check articles that have the current PEP
in them.
                ORDER BY similarity DESC -- order results descending to
find the most similar article
        ) s
        WHERE similarity > 0.80
        LIMIT 1; -- take the top, most similar article_id

   IF (art_id IS NOT NULL) THEN
                retMessg = 'DUPLICATE: ' || retMessg;
   ELSE -- New article to be added.

            -- full original JSON object is too large for indexing
            -- remove the 'body' and 'concepts' elements to reduce its
size.
            -- those elements will get saved in their own columns
            SELECT "ArticleJSON"::JSONB - 'body' - 'concepts'
                   INTO data;

            INSERT INTO finnova."Art" ("URI", "body", "concepts", "data")
                          VALUES (uriInt::INTEGER, jsonBody::TEXT,
concepts::JSONB, data::JSONB);

            -- find the art_id of the article that was just inserted
            SELECT a."ID"
                   INTO art_id
                   FROM finnova."Art" a
                  WHERE a."URI" = uriInt;
        END IF;

   -- Make an entry in the PEP_ART table for the pep and article
   -- if it was an article that was already in the database
   -- there is a chance that there is already an entry for it and the pep
   -- so on conflict do nothing.
   INSERT INTO finnova."PEP_ART" ("PEP_ID", "ART_ID")
        VALUES ("PEP_ID", art_id)
   ON CONFLICT DO NOTHING;

   RETURN retMessg;
EXCEPTION
   WHEN OTHERS
   THEN
      RAISE INFO 'Error Name:%', SQLERRM;
      RAISE INFO 'Error State:%', SQLSTATE;
      RETURN 'Error Will Robinson! ' || SQLERRM;
END;
$BODY$
        LANGUAGE plpgsql
        CALLED ON NULL INPUT
        VOLATILE
        EXTERNAL SECURITY INVOKER
        COST 100;
```

```
COMMENT ON FUNCTION "finnova"."addArticle" (IN PEP_ID int4, IN ArticleJSON
jsonb) IS 'This function is used for adding articles and associating a PEP
with them.  It first checks if there are any articles already in the
database that are at least 80% similar to the new one with the PEP in
them.  If articles are found that are mroe than 80% similar to the passed
in one, the passed in PEP is associated with the most simlar article  in
the PEP_ART table.
If no similar articles are found, the new one will be inserted into the
Art table.';

ALTER FUNCTION "finnova"."addArticle" (IN PEP_ID int4, IN ArticleJSON
jsonb) OWNER TO "FreiburgBill";
```

6.3 updatepep_art

Description

Annotation

Comment This function searches through all articles to ensure that all the PEPs in them are properly associtated with them in the PEP_ART table, along with how many times the PEP is in the article and the word positions of the PEP.

Owner FreiburgBill

Param Name	Param Type	Data Type
p_id	IN	int4

Create SQL

```
CREATE OR REPLACE FUNCTION "finnova"."updatepep_art" (IN p_id int4)

RETURNS text AS
$BODY$
DECLARE
   pep                   RECORD;
   art_id                INTEGER;
   b_art_id              INTEGER;
   pep_id                INTEGER;
   search_name           TEXT;
   lexeme_count          INTEGER;
   lexeme_positions      INTEGER [];
   _lexemes              tsvector;
   _searched_lexeme      tsvector;
   _occurences_pattern   TEXT;
   _occurences_list      TEXT;
BEGIN
   FOR art_id IN   SELECT a."ID" AS art_id
                     FROM finnova."Art" a
                    WHERE a."ID" > p_id
                ORDER BY art_id
   LOOP
      RAISE NOTICE 'Art_ID = % ', art_id;

      -- For every Article ID
      -- See if PEP_ID exists for it in PEP_ART
      FOR pep IN   SELECT p."ID" AS pep_id, p."lastname", p."searchname"
                     FROM finnova."MV_PEP" p
                ORDER BY pep_id
      LOOP
         --
         IF NOT EXISTS
               (SELECT *
```

```
                  FROM finnova."PEP_ART" pa
                 WHERE pa."ART_ID" = art_id AND pa."PEP_ID" = pep.pep_id)
         THEN
            -- The current PEP_ID is not associated with the current
ART_ID
            -- in the PEP_ART Table
            -- See if the current PEP name is in this article

            IF EXISTS
                  (SELECT *
                    FROM finnova."Art" a
                   WHERE     a."ID" = art_id
                         AND a.body LIKE ('%' || pep.searchname || '%'))
            THEN
               -- PEP exists in this article
               -- add a new PEP_ART entry for it

               INSERT INTO finnova."PEP_ART" ("PEP_ID", "ART_ID")
                    VALUES (pep.pep_id, art_id);

               RAISE NOTICE
               'SearchName = % Art_ID = % PEP_ID = %',
               pep.searchname, art_id, pep.pep_id;
            END IF;                                 -- PEP exists in this
article
         END IF;                 -- IF NOT EXISTS PEP_ID and ART_ID in
PEP_ART

                -- Now update the count of the current PEP  for the
current article
        -- in the PEP_ART table.
                  SELECT a.vector
                        INTO _lexemes
                        FROM finnova."Art" a
                   WHERE "ID" = art_id;

                  _searched_lexeme := strip (to_tsvector ('german',
pep.lastname));
                  _occurences_pattern := _searched_lexeme::TEXT || ':([0-
9,]+)';
                  _occurences_list := substring (_lexemes::TEXT,
_occurences_pattern);

                  SELECT count (a), array_agg (a::INT)
                        FROM regexp_split_to_table (_occurences_list,
',') a
                   WHERE _searched_lexeme::TEXT != ''        --
preventing false positives
                        INTO lexeme_count, lexeme_positions;

                  UPDATE finnova."PEP_ART" pa
                         SET count = lexeme_count, positions =
lexeme_positions
                   WHERE pa."ART_ID" = art_id AND pa."PEP_ID" =
pep.pep_id;

      END LOOP;                                                 -- through
PEPs


   END LOOP;                                                 -- through
Articles
```

```
   RETURN 'All done!';
END;
$BODY$
        LANGUAGE plpgsql
        CALLED ON NULL INPUT
        VOLATILE
        EXTERNAL SECURITY INVOKER
        COST 100;

COMMENT ON FUNCTION "finnova"."updatepep_art" (IN p_id int4) IS 'This
function searches through all articles to ensure that all the PEPs in them
are properly associtated with them in the PEP_ART table, along with how
many times the PEP is in the article and the word positions of the PEP.';

ALTER FUNCTION "finnova"."updatepep_art" (IN p_id int4) OWNER TO
"FreiburgBill";
```

6.4 find_pep_art

Description

Annotation

Comment This function is used to find the location of alert words in an article and the nearest PEP to that word. At the end it inserts the alert word, PEP, and article IDs into the ALERTWORDS_ART table and along with the word position of the alert word and PEP in the articles.

Owner FreiburgBill

Param Name	Param Type	Data Type
p_id	IN	int4

Create SQL

```
CREATE OR REPLACE FUNCTION "finnova"."find_pep_art" (IN p_id int4)
RETURNS text AS
$BODY$
DECLARE
   aword                RECORD;
   pep                  RECORD;
   pepart               RECORD;
   art                  RECORD;
   list                 RECORD;
   art_id               INTEGER;
   b_art_id             INTEGER;
   pep_id               INTEGER;
   closest_pep          INTEGER;
   closest_pos          INTEGER;
   aword_count          INTEGER;
   aword_positions      INTEGER [];
   search_name          TEXT;
   _lexemes             tsvector;
   _searched_lexeme     tsvector;
   _occurences_pattern  TEXT;
   _occurences_list     TEXT;
BEGIN
   -- get list of articles that have alert words in them
   FOR art IN   SELECT a."ID", a."vector"
                  FROM finnova."MV_ART" a
                 WHERE a."ID" > p_id AND a."alert" = TRUE
             ORDER BY art_id
   LOOP
```

```
      RAISE NOTICE 'Art_ID = % ', art."ID";

      -- For every Article ID
      -- Check every alert word
      FOR aword IN SELECT *
                     FROM finnova."ALERT_WORDS" aw
      LOOP
         --
         SELECT finnova.lexeme_occurrences (art.vector, aword."word",
'german')
           INTO list;

         RAISE NOTICE '>>>>>>>>>>>>>> List = % %', list, aword."word";

         _lexemes := art.vector;
         _searched_lexeme := strip (to_tsvector ('german', aword."word"));
         _occurences_pattern := _searched_lexeme::TEXT || ':([0-9,]+)';
         _occurences_list := substring (_lexemes::TEXT,
_occurences_pattern);

         SELECT count (a), array_agg (a::INT)
           FROM regexp_split_to_table (_occurences_list, ',') a
          WHERE _searched_lexeme::TEXT != ''     -- preventing false
positives
           INTO aword_count, aword_positions;


         FOR counter IN 1 .. aword_count
         LOOP
            RAISE NOTICE 'Counter: % %', aword_positions[counter],
aword_count;
           -- For the current alert word and its current position
           -- find the nearest PEP and its position.

          closest_pep := 0;  -- should end up with the ID of the closest
PEP
          closest_pos := 0;  -- should end up with the position of the
closest PEP

          -- Check all the PEPs that are associated with this article

                FOR pepart IN SELECT *
                                        FROM finnova."PEP_ART"
                                        WHERE "ART_ID" = art."ID"
                           LOOP
                               RAISE NOTICE 'PEP_ART PEP_ID: % %',
pepart."PEP_ID", pepart."positions";
                                    FOR pep_pos IN 1..pepart."count" -- loop
t
                           LOOP
                                    RAISE NOTICE 'PEP POS : %',
pepart."positions"[pep_pos];
                                    IF pepart."positions"[pep_pos] <
aword_positions[counter]
                                    AND pepart."positions"[pep_pos] >
closest_pos
                                    THEN
                                            -- if the current pep position
is before the alert word position
                                            -- and it is after the last
found closest position
```

```
                                        -- update the closest postion
and closest pep
                                        RAISE NOTICE 'Before Closest
PEP and POS: % %', closest_pep, closest_pos;
                                        closest_pep := pepart."PEP_ID";
                                    closest_pos :=
pepart."positions"[pep_pos];
                                    RAISE NOTICE 'After Closest PEP and
POS: % %', closest_pep, closest_pos;
                                END IF;
                            END LOOP; -- positions of pep in article


            END LOOP; -- pep_arts

                 IF closest_pep > 0 THEN
                    -- if a pep was found
                    -- Add an entry in the ALERTWORDS_ART table
                    -- for the values found above.

                    INSERT INTO finnova."ALERTWORDS_ART" ("AW_ID",
"PEP_ID", "AW_POS", "PEP_POS", "ART_ID")
                    VALUES (aword."ID", closest_pep,
aword_positions[counter], closest_pos, art."ID");
                END IF;

         END LOOP;  -- positions of alert words
      END LOOP;  -- alert words loop
   END LOOP;     -- article loop

   RETURN 'All done!';
END;
$BODY$
         LANGUAGE plpgsql
         CALLED ON NULL INPUT
         VOLATILE
         EXTERNAL SECURITY INVOKER
         COST 100;

COMMENT ON FUNCTION "finnova"."find_pep_art" (IN p_id int4) IS 'This
function is used to find the location of alert words in an article and the
nearest PEP to that word.  At the end it inserts the alert word, PEP, and
article IDs into the ALERTWORDS_ART table and along with the word position
of the alert word and PEP in the articles.';

ALTER FUNCTION "finnova"."find_pep_art" (IN p_id int4) OWNER TO
"FreiburgBill";
```

7. Views

7.1 MV_ALERT_WORDS

Description	
Annotation	
Comment	List the number of documents and number of times the alert words appear.
Owner	FreiburgBill

Create SQL

```
CREATE MATERIALIZED VIEW "finnova"."MV_ALERT_WORDS" AS
         SELECT ts_stat.word,
    ts_stat.ndoc,
    ts_stat.nentry
   FROM ts_stat('SELECT to_tsvector(''german'', body) from
finnova."Art"'::text) ts_stat(word, ndoc, nentry)
  WHERE (ts_stat.word IN ( SELECT
"substring"(replace((to_tsvector('german'::regconfig,
("ALERT_WORDS".word)::text))::text, ''':1'::text, ''::text), 2) AS
"substring"
           FROM finnova."ALERT_WORDS"));

COMMENT ON VIEW "finnova"."MV_ALERT_WORDS" IS 'List the number of
documents and number of times the alert words appear.';


ALTER TABLE "finnova"."MV_ALERT_WORDS" OWNER TO "FreiburgBill";
```

7.2 MV_ART

Description	
Annotation	
Comment	Materialized View of the Art (articles) table with the JSONB data column elements separated into their own columns and the body of the article as a tsvector column.
Owner	FreiburgBill

Create SQL

```
CREATE MATERIALIZED VIEW "finnova"."MV_ART" AS
        SELECT (a.data ->> 'uri'::text) AS uri,
    (a.data ->> 'title'::text) AS title,
    (a.data ->> 'url'::text) AS url,
    (a.data ->> 'image'::text) AS urltoimage,
    (a.data ->> 'dateTime'::text) AS publishedat,
    ((a.data -> 'source'::text) ->> 'title'::text) AS source,
    (((a.data #>> '{categories,0}'::text[]))::json ->> 'uri'::text) AS
category,
    (((a.data #>> '{authors,0}'::text[]))::json ->> 'name'::text) AS
author,
    (a.data ->> 'shares'::text) AS shares,
    a."ID",
    (lower(a.body) ~
similar_escape(((((('%(bestechung|geldwäsche|korruption|schmiergelder|waffen
handel|'::text ||
'drogenhandel|steuerhinterziehung|umweltverschmutzung|blutdiamanten|'::text
) || 'scheeballsysteme|insiderhandel|vorteilsnahme|ungetreue
geschäftsführung|'::text) || 'käuflich|schwarze
kasse|schwarzgeld|veruntreuung|unterschlagung|beeinflussung|'::text) ||
'Ponzi|betrug)%'::text), NULL::text)) AS alert,
```

```
    a.polarity,
    (a.data ->> 'location'::text) AS location,
    (a.data ->> 'evetURI'::text) AS eventuri,
    (a.data ->> 'storyURI'::text) AS storyuri,
    a.body,
    to_tsvector('german'::regconfig, a.body) AS vector
   FROM finnova."Art" a;

COMMENT ON VIEW "finnova"."MV_ART" IS 'Materialized View of the Art
(articles) table with the JSONB data column elements separated into their
own columns and the body of the article as a tsvector column.';


ALTER TABLE "finnova"."MV_ART" OWNER TO "FreiburgBill";
```

7.3 MV_PEP

Description	
Annotation	
Comment	Materialized View of the PEP table. Calculates age from year of birth, shows total number of articles per PEP, shows earliest date of an article per PEP, shows latest date of an article per PEP, shows the min, man, and avg sentiment polarity per PEP.
Owner	FreiburgBill

Create SQL

```
CREATE MATERIALIZED VIEW "finnova"."MV_PEP" AS
         SELECT p."ID",
        CASE
            WHEN (substr((p."Name")::text, 1, 3) = 'de '::text) THEN
btrim("substring"((p."Name")::text, 'de [^ ]*'::text))
            ELSE btrim(substr((p."Name")::text, 1,
"position"((p."Name")::text, ' '::text)))
        END AS lastname,
        CASE
            WHEN (substr((p."Name")::text, 1, 3) = 'de '::text) THEN
btrim("substring"((p."Name")::text, '(?:\S+ ){2}(.*)'::text))
            ELSE btrim(substr((p."Name")::text,
"position"((p."Name")::text, ' '::text)))
        END AS firstname,
        CASE
            WHEN (substr((p."Name")::text, 1, 3) = 'de '::text) THEN
(("substring"((p."Name")::text, '(?:\S+ ){2}(.*)'::text) || ' '::text) ||
"substring"((p."Name")::text, 'de [^ ]*'::text))
            ELSE ((btrim(substr((p."Name")::text,
"position"((p."Name")::text, ' '::text))) || ' '::text) ||
btrim(substr((p."Name")::text, 1, "position"((p."Name")::text, '
'::text))))
        END AS searchname,
    p."Gender",
    p."Language",
    p."Year",
    (date_part('year'::text, CURRENT_DATE) - (p."Year")::double precision)
AS age,
    s.num_articles,
    s.earliest,
    s.latest,
    s.min,
```

```
    s.max,
    s.avg,
    s.num_alerts
   FROM (finnova."PEP" p
     LEFT JOIN ( SELECT pa."PEP_ID",
            count(*) AS num_articles,
            ( SELECT count(*) AS count
                   FROM finnova."V_PEP_ART2" v
                  WHERE ((v.alert = true) AND (v."PEP_ID" = pa."PEP_ID")))
AS num_alerts,
            min((a.data ->> 'dateTime'::text)) AS earliest,
            max((a.data ->> 'dateTime'::text)) AS latest,
            min(a.polarity) AS min,
            max(a.polarity) AS max,
            avg(a.polarity) AS avg
           FROM (finnova."Art" a
             LEFT JOIN finnova."PEP_ART" pa ON ((a."ID" = pa."ART_ID")))
          GROUP BY pa."PEP_ID") s ON ((p."ID" = s."PEP_ID")));

COMMENT ON VIEW "finnova"."MV_PEP" IS 'Materialized View of the PEP table.
Calculates age from year of birth,
shows total number of articles per PEP,
shows earliest date of an article per PEP,
shows latest date of an article per PEP,
shows the min, man, and avg sentiment polarity per PEP.';


ALTER TABLE "finnova"."MV_PEP" OWNER TO "FreiburgBill";
```

7.4 MV_Sources

Description	
Annotation	
Comment	
Owner	FreiburgBill

Create SQL

```
CREATE MATERIALIZED VIEW "finnova"."MV_Sources" AS
         SELECT v.source,
    count(v.article_id) AS num_articles,
    min(v.publishedat) AS earliest,
    max(v.publishedat) AS latest,
    min(v.polarity) AS min,
    max(v.polarity) AS max,
    avg(v.polarity) AS avg,
    count(
        CASE
            WHEN v.alert THEN 1
            ELSE NULL::integer
        END) AS num_alerts
   FROM finnova."V_Art" v
  GROUP BY v.source;

ALTER TABLE "finnova"."MV_Sources" OWNER TO "FreiburgBill";
```

7.5 MV_WORD_PEP_ART

Description

Annotation	
Comment	Shows alert words and their associated PEPs position in an article along with the article title, polarity, and url.
Owner	FreiburgBill

Create SQL

```
CREATE MATERIALIZED VIEW "finnova"."MV_WORD_PEP_ART" AS
         SELECT aw.word,
    awa."AW_POS",
    p."ID" AS pep_id,
    p.searchname,
    awa."PEP_POS",
    (awa."AW_POS" - awa."PEP_POS") AS score,
    a.title,
    a.publishedat,
    a.polarity,
    a."ID" AS art_id,
    a.url,
    a.category,
    a.author,
    a.shares
   FROM (((finnova."ALERTWORDS_ART" awa
     LEFT JOIN finnova."ALERT_WORDS" aw ON ((awa."AW_ID" = aw."ID")))
     LEFT JOIN finnova."MV_PEP" p ON ((awa."PEP_ID" = p."ID")))
     LEFT JOIN finnova."MV_ART" a ON ((awa."ART_ID" = a."ID")));

COMMENT ON VIEW "finnova"."MV_WORD_PEP_ART" IS 'Shows alert words and
their associated PEPs position in an article along with the article title,
polarity, and url.';


ALTER TABLE "finnova"."MV_WORD_PEP_ART" OWNER TO "FreiburgBill";
```

7.6 V_PEP_ART2

Description	
Annotation	
Comment	
Owner	FreiburgBill

Create SQL

```
CREATE OR REPLACE VIEW "finnova"."V_PEP_ART2" AS
         SELECT p."Name",
    (a.data ->> 'url'::text) AS url,
    (a.data ->> 'title'::text) AS title,
    (lower(a.body) ~
similar_escape(((((('%(bestechung|geldwäsche|korruption|schmiergelder|waffen
handel|'::text ||
'drogenhandel|steuerhinterziehung|umweltverschmutzung|blutdiamanten|'::text
) || 'scheeballsysteme|insiderhandel|vorteilsnahme|ungetreue
geschäftsführung|'::text) || 'käuflich|schwarze
kasse|schwarzgeld|veruntreuung|unterschlagung|beeinflussung|'::text) ||
'Ponzi|betrug)%'::text), NULL::text)) AS alert,
    ((a.data -> 'source'::text) ->> 'title'::text) AS source,
    (a.data ->> 'dateTime'::text) AS publishedat,
    pa."PEP_ID",
    pa."ART_ID",
    a.data
   FROM ((finnova."PEP" p
     LEFT JOIN finnova."PEP_ART" pa ON ((p."ID" = pa."PEP_ID")))
```

```
    LEFT JOIN finnova."Art" a ON ((pa."ART_ID" = a."ID")));

ALTER TABLE "finnova"."V_PEP_ART2" OWNER TO "FreiburgBill";
```