Physical Data Diagram

Finnova Schema

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

Table of Contents

* [1. Diagram Information](" \l "ModelInfo)
  + [1.1. Basic Information](#BasicInfo)
  + [1.2. Diagram Description](#ModelDesc)
  + [1.3. Diagram Annotation](#ModelAnn)
  + [1.4. ER Diagram](#ModelERD)
* [2. Database](#Database)
* [3. Domains](#Domains)
* [4. Tables](#Tables)
  + [4.1 ALERTWORDS\_ART](#ALERTWORDS_ART)
    - [4.1.1 Columns](#ALERTWORDS_ART_Columns)
    - [4.1.2 Constraints](#ALERTWORDS_ART_Constraints)
    - [4.1.3 Indexes](#ALERTWORDS_ART_Indexes)
    - [4.1.4 Rules](#ALERTWORDS_ART_Rules)
    - [4.1.5 Triggers](#ALERTWORDS_ART_Triggers)
  + [4.2 ALERT\_WORDS](#ALERT_WORDS)
    - [4.2.1 Columns](#ALERT_WORDS_Columns)
    - [4.2.2 Constraints](#ALERT_WORDS_Constraints)
    - [4.2.3 Indexes](#ALERT_WORDS_Indexes)
    - [4.2.4 Rules](#ALERT_WORDS_Rules)
    - [4.2.5 Triggers](#ALERT_WORDS_Triggers)
  + [4.3 Art](#Art)
    - [4.3.1 Columns](#Art_Columns)
    - [4.3.2 Constraints](#Art_Constraints)
    - [4.3.3 Indexes](#Art_Indexes)
    - [4.3.4 Rules](#Art_Rules)
    - [4.3.5 Triggers](#Art_Triggers)
  + [4.4 PEP](#PEP)
    - [4.4.1 Columns](#PEP_Columns)
    - [4.4.2 Constraints](#PEP_Constraints)
    - [4.4.3 Indexes](#PEP_Indexes)
    - [4.4.4 Rules](#PEP_Rules)
    - [4.4.5 Triggers](#PEP_Triggers)
  + [4.5 PEP\_ART](#PEP_ART)
    - [4.5.1 Columns](#PEP_ART_Columns)
    - [4.5.2 Constraints](#PEP_ART_Constraints)
    - [4.5.3 Indexes](#PEP_ART_Indexes)
    - [4.5.4 Rules](#PEP_ART_Rules)
    - [4.5.5 Triggers](#PEP_ART_Triggers)
* [5. References](#References)
  + [5.1 FK\_ALERTWORDS\_ART\_AW](#FK_ALERTWORDS_ART_AW)
  + [5.2 FK\_ALERTWORDS\_ART\_PEP](#FK_ALERTWORDS_ART_PEP)
  + [5.3 FK\_ALERTWORDS\_ART\_ART](#FK_ALERTWORDS_ART_ART)
  + [5.4 FK\_PEP\_ART\_1](#FK_PEP_ART_1)
  + [5.5 FK\_PEP\_ART\_2](#FK_PEP_ART_2)
* [6. Stored Procedures](#StoredProces)
  + [6.1 lexeme\_occurrences](#lexeme_occurrences)
  + [6.2 addArticle](#addArticle)
  + [6.3 updatepep\_art](#updatepep_art)
  + [6.4 find\_pep\_art](#find_pep_art)
* [7. Views](#Views)
  + [7.1 MV\_ALERT\_WORDS](#MV_ALERT_WORDS)
  + [7.2 MV\_ART](#MV_ART)
  + [7.3 MV\_PEP](#MV_PEP)
  + [7.4 MV\_Sources](#MV_Sources)
  + [7.5 MV\_WORD\_PEP\_ART](#MV_WORD_PEP_ART)
  + [7.6 V\_PEP\_ART2](#V_PEP_ART2)

**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***

**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|waffenhandel|'::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|waffenhandel|'::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"; |