graph_CVE_ORG

```sql
SELECT *FROM test.org


DROP TABLE IF EXISTS  test.graph_cve_org_tmp;
CREATE TABLE test.graph_cve_org_tmp AS
WITH wt_cve_org_extract AS
(
SELECT
    t.cve_id AS id,
    COALESCE(affecteds.value ->> 'vendor', '*') AS vendor,
    COALESCE(affecteds.value ->> 'product', '*') AS product,
    COALESCE(affecteds.value ->> 'packageName', '*') AS modules_packageName,
    affecteds.value ->> 'platforms' AS platforms,
    COALESCE(affecteds.value ->> 'repo', '') AS repo,
    affecteds.value ->> 'collectionURL' AS collectionURL,
    CASE WHEN upper(COALESCE(affected_versions.value ->> 'versionType','*'))  IN ('NPM', 'CRATES.IO', 'PYPI', '
--    affected_versions.value ->> 'version' AS affected_version,
--    affected_versions.value ->> 'lessThan' AS affected_lessThan,
--    affected_versions.value ->> 'lessThanOrEqual' AS affected_lessThanOrEqual,
    affecteds.value ->> 'defaultStatus' AS affected_defaultStatus ,
    affected_versions.value ->> 'status' AS affected_status,
    affected_versions.value AS affected_versions,
    t.cve_msg -> 'containers' -> 'cna' -> 'source' ->> 'discovery' AS source_discovery,
    t.cve_msg -> 'containers' -> 'cna' -> 'x_legacyV4Record' -> 'CVE_data_meta' ->> 'ASSIGNER' AS source_v4_ass
    t.cve_msg -> 'cveMetadata' ->> 'assignerShortName' AS source_assignerShortName, --identify
    t.cve_msg -> 'containers' -> 'cna' ->> 'title' AS desc_title,
--    t.cve_msg -> 'containers' -> 'cna' -> 'descriptions' -> 0 ->> 'lang' AS desc_details_lang,
    t.cve_msg -> 'containers' -> 'cna' -> 'descriptions' -> 0 ->> 'value' AS desc_details_value,
    problemtype_descs.value -> 'descriptions' -> 0 ->> 'type'  AS problemtype_descs_type,
    problemtype_descs.value -> 'descriptions' -> 0 ->> 'cweId'  AS problemtype_descs_cweId,
    problemtype_descs.value -> 'descriptions' -> 0 ->> 'description'  AS problemtype_descs_detail,
--    problemtype_descs.value -> 'descriptions' -> 0 ->> 'references'  AS problemtype_references,
    t.cve_msg -> 'containers' -> 'cna' -> 'metrics' AS severity,
--    metrics.value -> 'cvssV4_0'AS severity_cvssV4_0,
--    metrics.value -> 'cvssV3_1'AS severity_cvssV3_1,
--    metrics.value -> 'cvssV3_0'AS severity_cvssV3_0,
--    metrics.value -> 'cvssV2_0'AS severity_cvssV2_0,
    t.cve_msg -> 'cveMetadata' ->> 'datePublished' AS time_info_published,
    t.cve_msg -> 'cveMetadata' ->> 'dateUpdated' AS time_info_lastModified,
    t.cve_msg -> 'cveMetadata' ->> 'datePublished' AS time_info_firstpublished,
    t.cve_msg -> 'containers' -> 'cna' -> 'impacts' AS impacts,
    solutions.value ->> 'lang' AS solutions_lang, --en 或者 eng
    solutions.value ->> 'value' AS solutions_value,
    workarounds.value AS solutions_workarounds_val,
    exploits.value ->> 'value' AS exploits_val,
    refs.value ->> 'url' AS ref_url,
    refs.value ->> 'name' AS ref_name,
    refs.value ->> 'tags' AS ref_tag
FROM test.ods_cve_org_cvelist_source_msg t
```

```sql
LEFT JOIN jsonb_array_elements(t.cve_msg -> 'containers' -> 'cna' -> 'affected') affecteds ON 1=1
LEFT JOIN jsonb_array_elements(affecteds.value -> 'versions') affected_versions ON 1=1
LEFT JOIN jsonb_array_elements(t.cve_msg -> 'containers' -> 'cna' -> 'problemTypes' ) problemtype_descs ON 1=1
--LEFT JOIN jsonb_array_elements(t.cve_msg -> 'containers' -> 'cna' -> 'metrics' ) metrics ON 1=1
LEFT JOIN jsonb_array_elements(t.cve_msg -> 'containers' -> 'cna' -> 'solutions') solutions ON 1=1
LEFT JOIN jsonb_array_elements(t.cve_msg -> 'containers' -> 'cna' -> 'workarounds') workarounds ON 1=1
LEFT JOIN jsonb_array_elements(t.cve_msg -> 'containers' -> 'cna' -> 'exploits') exploits ON 1=1
LEFT JOIN jsonb_array_elements(t.cve_msg -> 'containers' -> 'cna' -> 'references') refs ON 1=1
)
SELECT *
FROM
(
SELECT oe.*, CASE WHEN oe.vendor = 'n/a' THEN '*' ELSE oe.vendor END AS vendor_op,
        CASE WHEN oe.product = 'n/a' THEN '*' ELSE oe.product END AS product_op,
        CASE WHEN oe.modules_packageName = 'n/a' THEN '*' ELSE oe.modules_packageName END AS modules_packageNam
--SELECT DISTINCT affected_status
FROM wt_cve_org_extract oe
)tmp_oe
--WHERE tmp_oe.vendor_op||tmp_oe.product_op||tmp_oe.modules_packageName_op||tmp_oe.ecosystem <> '****';

SELECT *FROM graph_cve_org_tmp WHERE id = 'CVE-2007-10002'


DROP TABLE IF EXISTS test.graph_node_vul_cve_org;
CREATE TABLE test.graph_node_vul_cve_org AS
WITH vul_node_tmp AS
(
SELECT t.id  ,
      jsonb_build_object('discovery',  t.source_discovery, 'identifier',  COALESCE(t.source_v4_assigner, t.sour
     jsonb_build_object('title', t.desc_title, 'details', t.desc_details_value) AS description,
     jsonb_build_object('type', NULL, 'cweId', CASE WHEN t.problemtype_descs_type = 'CWE' THEN t.problemtype_de
                      'description', CASE WHEN t.problemtype_descs_type = 'CWE' THEN t.problemtype_descs_detai
     t.severity,
     jsonb_build_object('published', t.time_info_published, 'lastModified', t.time_info_lastmodified, 'datePubl
     jsonb_build_object('solutions', t.solutions_value, 'workarounds', t.solutions_workarounds_val) AS solution
     jsonb_build_object('exploitable', NULL, 'exploits',  t.exploits_val, 'exploit_url', NULL, 'exploitabilityS
     jsonb_build_object('PoC_available', NULL, ' PoC_url', NULL ) AS PoC_info,
     jsonb_build_object('patch_available', NULL, 'patch_url', null) AS  patch_info,
     jsonb_build_object('report_status', null) ,
     t.vendor_op, t.product_op, t.modules_packageName_op, t.ecosystem
FROM test.graph_cve_org_tmp t
)
SELECT vnt.id, '[]' AS aliases, vnt.SOURCE, vnt.description, jsonb_agg(DISTINCT vnt.weaknesses) AS weaknesses,
      jsonb_build_object('solution_info', jsonb_agg(DISTINCT vnt.solution_info) ,
                         'exploit_info', jsonb_agg(DISTINCT vnt.exploit_info),
                          'PoC_info', jsonb_build_object('PoC_available', NULL, ' PoC_url', NULL ),
                          'patch_info', jsonb_build_object('patch_available', NULL, 'patch_url', null),
                          'report_status', NULL ) AS status,
      vnt.vendor_op, vnt.product_op, vnt.modules_packageName_op, vnt.ecosystem
FROM vul_node_tmp vnt
GROUP BY vnt.id, vnt.SOURCE, vnt.description, vnt.severity, vnt.time_info, vnt.vendor_op, vnt.product_op, vnt.m



DELETE FROM test.dws_graph_node_vul WHERE vul_source = 'CVE';
ALTER SEQUENCE cve_graph_seq RESTART START WITH 1;
INSERT INTO test.dws_graph_node_vul
SELECT nextval('cve_graph_seq') AS seq, tmp.*
FROM
```

```
(
SELECT  DISTINCT  id ,  aliases , "source", description ,weaknesses, severity, time_info,  status, 'CVE' AS vul
)tmp ;







SELECT count(*)
FROM test.dws_graph_node_vul t
WHERE t.vul_source = 'CVE'
GROUP BY t.id
HAVING count(*) > 1




LOAD CSV WITH HEADERS FROM 'file:///node_vul_${vul_source}_${num}.csv' AS row
MERGE (n:Vuln_${vul_source} {id: row.id})
SET n.aliases = row.aliases,
    n.source = row.SOURCE,
    n.description = row.description,
    n.weaknesses = row.weaknesses,
    n.severity = row.severity,
    n.time_info = row.time_info,
    n.status = row.status;


DROP TABLE IF EXISTS test.graph_node_component;
ALTER SEQUENCE cve_graph_seq RESTART START WITH 1;
CREATE TABLE test.graph_node_component AS
SELECT nextval('cve_graph_seq') AS seq, tmp.*
FROM
(
SELECT DISTINCT  vnt.id, vnt.product_op AS component_name,
      vnt.vendor_op AS vendor,
      vnt.modules_packageName_op AS  package_name,
      vnt.ecosystem AS ecosystem,
      vnt.affected_defaultstatus ,
      CASE WHEN vnt.affected_status  = 'affected'  THEN vnt.affected_versions  END AS affected_versions,
      CASE WHEN vnt.affected_status  = 'unaffected'  THEN vnt.affected_versions  END AS unaffected_versions,
      vnt.platforms,
      vnt.collectionurl,
      vnt.repo AS repo_url
FROM test.graph_cve_org_tmp vnt
WHERE  vnt.vendor_op||vnt.product_op||vnt.modules_packageName_op||vnt.ecosystem <> '****') tmp ;

\copy (SELECT DISTINCT t.component_name, t.vendor , t.package_name, t.ecosystem FROM graph_node_component t) to

\copy (SELECT *  FROM graph_node_component t) to 'r_component_cve.csv' with  (delimiter ',', FORCE_QUOTE *, for


--SELECT DISTINCT t.id, t.component_name, t.vendor , t.package_name, t.ecosystem, t.affected_msg  FROM graph_no

LOAD CSV WITH HEADERS FROM 'file:///n_component_cve.csv' AS row
CREATE (n:affected_components {component_name: row.component_name, vendor: row.vendor,  package_name: row.packa
                          ecosystem: row.ecosystem});
create index for (n:affected_components) on (n.component_name, n.vendor, n.package_name, n.ecosystem);

SELECT *FROM graph_node_component LIMIT 10

LOAD CSV WITH HEADERS FROM 'file:///r_component_cve_10w_4.csv' AS row
             MATCH (cve:Vuln_CVE {id: row.id})
             MATCH (lib:affected_components {component_name: row.component_name, vendor: row.vendor, package_na
             MERGE (cve)-[r:AFFECTS{repo_url: COALESCE(row.repo,''),  platform: COALESCE(row.platforms,''), col




DROP TABLE IF EXISTS test.t_tmp_graph_node_refs_CVE;
CREATE TABLE test.t_tmp_graph_node_refs_CVE AS
SELECT DISTINCT cot.id , cot.ref_url , cot.ref_name , cot.ref_tag
FROM test.graph_cve_org_tmp cot
WHERE cot.ref_url IS NOT NULL;

DELETE FROM test.dws_graph_node_refs WHERE vul_source = 'CVE';
ALTER SEQUENCE cve_graph_seq RESTART START WITH 1;
INSERT INTO test.dws_graph_node_refs
  SELECT nextval('cve_graph_seq') AS seq, tmp.*
  FROM (
    SELECT   DISTINCT ref_url, 'CVE' AS source  FROM   test.t_tmp_graph_node_refs_CVE
  )tmp;

 DELETE FROM test.dws_graph_relationships_refs WHERE vul_source = 'CVE';
ALTER SEQUENCE cve_graph_seq RESTART START WITH 1;
INSERT INTO test.dws_graph_relationships_refs
  SELECT nextval('cve_graph_seq') AS seq, tmp.*
  FROM (
    SELECT   DISTINCT  id, ref_url , ref_tag AS tags, ref_name AS ref_desc , 'CVE' AS vul_source  FROM   test.t
  )tmp;
bash gen_graph_data.sh "CVE" "relationships" "refs"
bash neo4j_relationships_refs_load.sh "CVE" "0"


SELECT *FROM test.t_tmp_graph_node_refs_CVE LIMIT 10


DROP TABLE IF EXISTS test.t_tmp_graph_node_cwe_CVE;
CREATE TABLE test.t_tmp_graph_node_cwe_CVE AS
SELECT *
FROM
(
SELECT DISTINCT t.id , t.problemtype_descs_cweid  AS cwe_id, t.problemtype_descs_detail AS cwe_desc, problemtyp
FROM test.graph_cve_org_tmp t
WHERE t.problemtype_descs_type = 'CWE'
)t
WHERE t.cwe_id IS NOT NULL ;

SELECT *FROM test.t_tmp_graph_node_cwe_CVE  WHERE cwe_id IS NULL

DELETE FROM test.dws_graph_node_cwe WHERE vul_source = 'CVE';
ALTER SEQUENCE cve_graph_seq RESTART START WITH 1;
INSERT INTO test.dws_graph_node_cwe
  SELECT nextval('cve_graph_seq') AS seq, tmp.*
  FROM (
    SELECT   DISTINCT cwe_id, 'CVE'   FROM   test.t_tmp_graph_node_cwe_CVE
  )tmp;
bash gen_graph_data.sh "CVE" "node" "cwe"
bash neo4j_cwe_node_load.sh "CVE" "0"

SELECT *FROM test.t_tmp_graph_node_cwe_CVE LIMIT 10


DELETE FROM test.dws_graph_relationships_cwe WHERE vul_source = 'CVE';
ALTER SEQUENCE cve_graph_seq RESTART START WITH 1;
INSERT INTO test.dws_graph_relationships_cwe
  SELECT nextval('cve_graph_seq') AS seq, tmp.*
  FROM (
    SELECT   DISTINCT   id, cwe_id , '' AS cwe_type, cwe_desc , 'CVE' AS vul_source  FROM   test.t_tmp_graph_no
  )tmp;
bash gen_graph_data.sh "CVE" "relationships" "cwe"
bash neo4j_relationships_cwe_load.sh "CVE" "0"

            SELECT count(*) FROM test.cve_org ods_cve_org_cvelist_source_msg
```