The Pitfalls Within Duality Views

Philipp Salvisberg
11th March 2026

GRiSSELBAV
simple & accurate IT solutions.

Timer?

Welcome

Philipp Salvisberg
Founder, Owner and CEO of Grisselbav GmbH

- Database-centric development

- Open-Source development

- Simple & accurate solutions

philipp.salvisberg@grisselbav.com

https://www.salvis.com/blog/

What is a DV?

DML via JSON-Relational Duality View (DV)

```json
{
  "_id" : 50,
  "dname" : "MI6",
  "loc" : "LONDON",
  "secret" : true,
  "emps" :
  [
    {
      "empno" : 7,
      "ename" : "BOND",
      "job" : "AGENT",
      "mgr" : 1,
      "hiredate" : "1950-01-01T00:00:00",
      "sal" : 500,
      "tools" : ["Knife", "Garrote Watch",
                 "Walther PPK"]
    },
    {
      "empno" : 1,
      "ename" : "M",
      "job" : "MANAGER",
      "hiredate" : "1940-01-01T00:00:00",
      "sal" : 1000,
      "comm" : 8000
    }
  ]
}
```

DV

DEPT

DEPTNO	DNAME	LOC	EXT
50	MI6	LONDON	{"secret":true}

EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	EXT
1	M	MANAGER		1940-01-01	1000	8000	50	
7	BOND	AGENT	1	1950-01-01	500		50	{"tools":["Knife","Garrote Watch","Walther PPK"]}

```sql
create or replace json duality view dept_dv as
select json {
        '_id': deptno,
        dname,
        loc,
        ext as flex,
        'emps':
            (
            select json_arrayagg(
                    json {
                        emp.empno,
                        emp.ename,
                        emp.job,
                        unnest
                            (
                            select json {
                                    'mgr'    : mgr.empno with nocheck,
                                    'mgrname': mgr.ename with nocheck
                                }
                                from emp mgr
                                where mgr.empno = emp.mgr
                            ),
                        emp.hiredate,
                        emp.sal,
                        emp.comm,
                        ext as flex
                    }
                )
                from emp with insert update delete
                where emp.deptno = dept.deptno
            )
    }
    from dept with insert update delete;
```

Annotations used:
flex, unnest, nocheck,
link (join condition),
insert, update, delete

Applicable
in DVs only

DV using GraphQL

```
create or replace json duality view dept_dv as
dept @insert @update @delete
{
    _id: deptno
    dname
    loc
    ext @flex
    emps: emp @insert @update @delete
        {
            empno
            ename
            job
            emp @unnest @link(from: [mgr])
                {
                    mgr     : empno @nocheck
                    mgrname: ename @nocheck
                }
            hiredate
            sal
            comm
            ext @flex
        }
};
```

All annotations (directives) start with '@'

Default Join

Shorter

Looks similar to the result

Motivation

dbLinter – JSON and DVs

Duality View to Export/Import Tenants
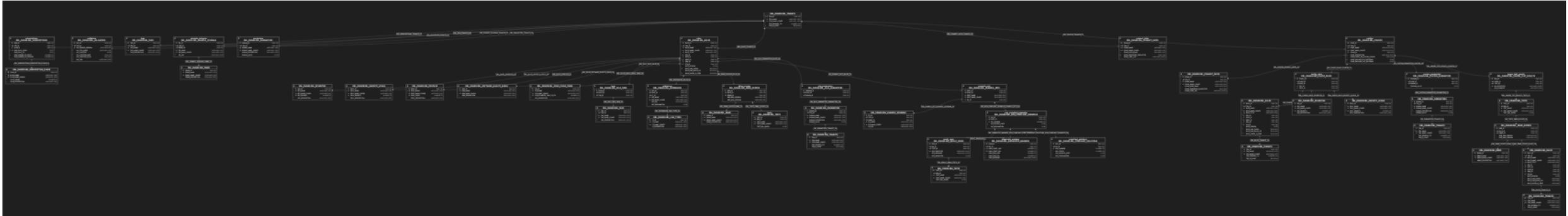
```
create or replace json duality view dbl_tenants_expimp_dv as
dbl_tenants @insert @update @delete
{
    _id: ten_id
    ten_name
    ten_session_ttl
    ten_is_core
    subscriptions: dbl_subscriptions @insert @update @delete
        { -- ins/upd/del requires Product-Admin role
        sub_id
        dbl_subscription_plans @unnest
            {
            plan_id @nocheck
            plan_name @nocheck
            }
        sub_valid_from
        sub_valid_to
        sub_number_of_seats
        sub_number_of_configs
        }
    validators: dbl_validators @insert @update @delete
        {
        val_id
        val_parser_version
        val_file_name
        val_content
        val_content_md5
        val_content_sha1
        val_url
        }
    tags: dbl_tags @insert @update @delete
        {
        tag_id
        tag_name
        tag_description
        }
    example_schemas: dbl_example_schemas
                @insert @update @delete
        {
        es_id
        dbl_dbms @unnest
            {
            dbms_id @nocheck
            dbms_name @nocheck
            }
        es_name
        es_url
        }
    parameters: dbl_parameters @insert @update @delete
        {
        param_id
        param_name
        param_description
        param_value
        }
    rules: dbl_rules @insert @update @delete
        {
        rule_id
        rule_name
        rule_title
        dbl_severities @unnest
            {
            sev_id @nocheck
            sev_name @nocheck
            }
        dbl_severity_levels @unnest
            {
            sevl_id @nocheck
            sevl_name @nocheck
            }
        dbl_profiles @unnest
            {
            prof_id @nocheck
            prof_name @nocheck
            }
```
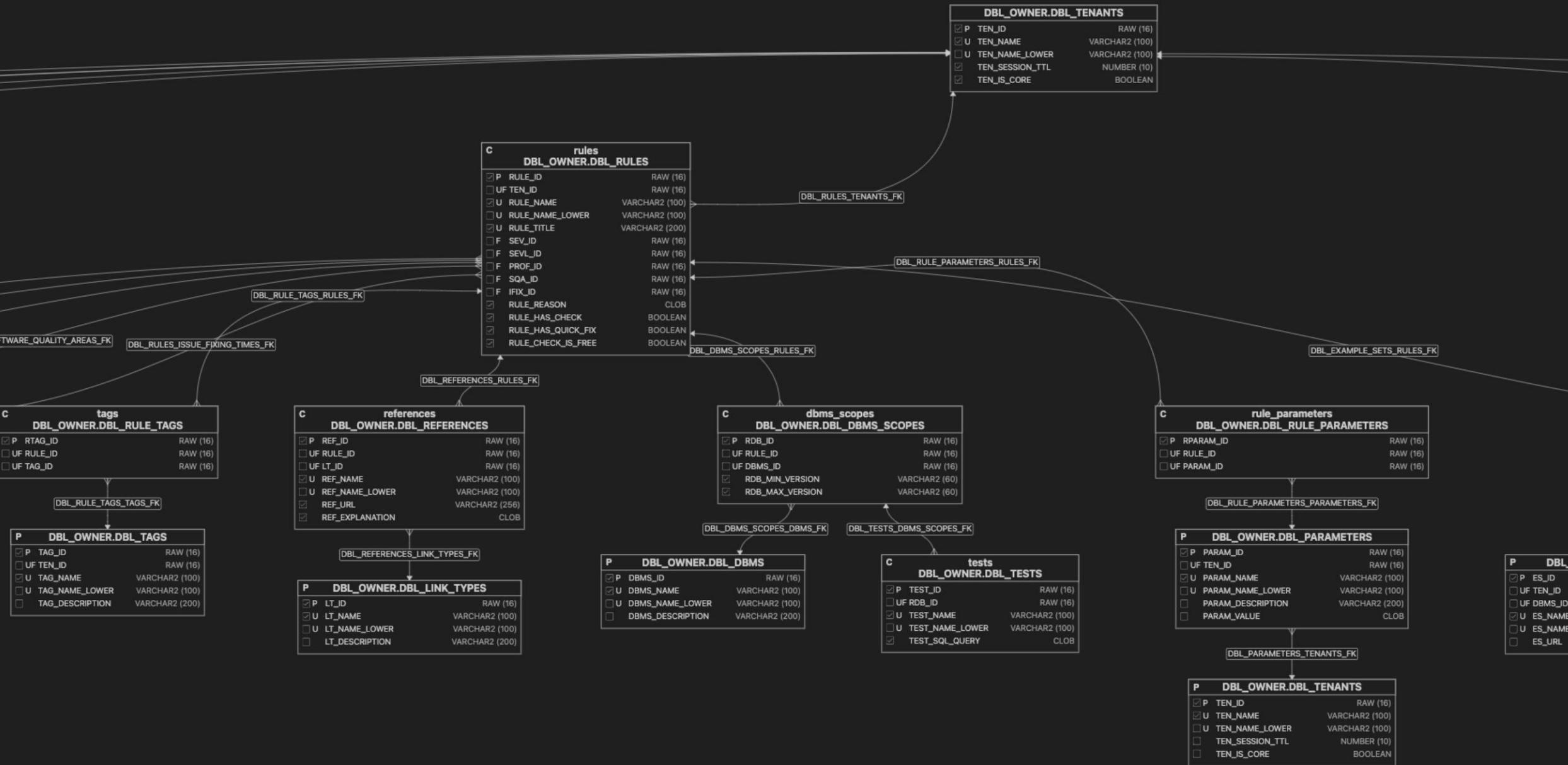
```
        dbl_software_quality_areas @unnest
            {
            sqa_id @nocheck
            sqa_name @nocheck
            }
        dbl_issue_fixing_times @unnest
            {
            ifix_id @nocheck
            ifix_name @nocheck
            }
        rule_reason
        rule_has_check
        rule_has_quick_fix
        rule_check_is_free
        tags: dbl_rule_tags @insert @update @delete
            {
            rtag_id
            dbl_tags @unnest
                {
                tag_id @nocheck
                tag_name @nocheck
                }
            }
        references: dbl_references
                @insert @update @delete
            {
            ref_id
            dbl_link_types @unnest
                {
                lt_id @nocheck
                lt_name @nocheck
                }
            ref_name
            ref_url
            ref_explanation
            }
        dbms_scopes: dbl_dbms_scopes
                @insert @update @delete
            {
            rdb_id
            dbl_dbms @unnest
                {
                dbms_id @nocheck
                dbms_name @nocheck
                }
            rdb_min_version
            rdb_max_version
            tests: dbl_tests @insert @update @delete
                {
                test_id
                test_name
                test_sql_query
                }
            }
        rule_parameters: dbl_rule_parameters
                @insert @update @delete
            {
            rparam_id
            dbl_parameters @unnest
                {
                param_id @nocheck
                param_name @nocheck
                dbl_tenants @unnest
                    {
                    ten_id @nocheck
                    ten_name @nocheck
                    }
                }
            }
        example_sets: dbl_example_sets
                @insert @update @delete
            {
            eset_id
```

```
            eset_name
            dbl_example_schemas @unnest
                {
                es_id @nocheck
                es_name @nocheck
                }
            non_compliant_examples:
            dbl_non_compliant_examples @insert @update @delete
                {
                nc_id
                nc_number
                nc_source_code
                nc_explanation
                result_rows: dbl_result_rows
                        @insert @update @delete
                    {
                    res_id
                    dbl_tests @unnest
                        {
                        test_id @nocheck
                        test_name @nocheck
                        }
                    res_identifier,
                    res_message,
                    res_migration
                    }
                diagnostic_markers:
                dbl_diagnostic_markers @insert @update @delete
                    {
                    diag_id
                    diag_start_line
                    diag_start_col
                    diag_end_line
                    diag_end_col
                    diag_message
                    }
                compliant_solutions:
                dbl_compliant_solutions @insert @update @delete
                    {
                    sol_id
                    sol_number
                    sol_stars
                    sol_source_code
                    sol_explanation
                    }
                }
            }
    connect_infos: dbl_connect_infos @insert @update @delete
        {
        conn_id
        conn_name
        conn_user_name
        conn_password_encrypted
        conn_jdbc_url
        }
    configs: dbl_configs @insert @update @delete
        {
        conf_id
        conf_name
        dbl_connect_infos @unnest
            {
            conn_id @nocheck
            conn_name @nocheck
            }
        conf_description
        conf_check_on_syntax_error
        conf_include_file_patterns
        conf_exclude_file_patterns
        config_rules: dbl_config_rules
                @insert @update @delete
            {
            crule_id
            dbl_rules @unnest
                {
```

```
                rule_id @nocheck
                rule_name @nocheck
                dbl_tenants @unnest
                    {
                    ten_id @nocheck
                    ten_name @nocheck
                    }
                }
            dbl_severities @unnest
                {
                sev_id @nocheck
                sev_name @nocheck
                }
            dbl_severity_levels @unnest
                {
                sevl_id @nocheck
                sevl_name @nocheck
                }
            custom_parameters: dbl_custom_parameters
                        @insert @update @delete
                {
                cparam_id
                dbl_parameters @unnest
                    {
                    param_id @nocheck
                    param_name @nocheck
                    dbl_tenants @unnest
                        {
                        ten_id @nocheck
                        ten_name @nocheck
                        }
                    }
                cparam_value
                }
            ignore_test_results: dbl_ignore_test_results
                        @insert @update @delete
                {
                ign_id
                dbl_tests @unnest
                    {
                    test_id @nocheck
                    test_name @nocheck
                    dbl_dbms_scopes @unnest
                        {
                        rdb_id @nocheck
                        dbl_dbms @unnest
                            {
                            dbms_id @nocheck
                            dbms_name @nocheck
                            }
                        dbl_rules @unnest
                            {
                            rule_id @nocheck
                            rule_name @nocheck
                            dbl_tenants @unnest
                                {
                                ten_id @nocheck
                                ten_name @nocheck
                                }
                            }
                        }
                    }
                ign_identifier
                ign_comment
                }
            }
        }
};
```

Duality View Based on 48 Tables

DBL_OWNER.DBL_TENANTS

P	TEN_ID	RAW (16)
U	TEN_NAME	VARCHAR2 (100)
U	TEN_NAME_LOWER	VARCHAR2 (100)
	TEN_SESSION_TTL	NUMBER (10)
	TEN_IS_CORE	BOOLEAN

**C rules
DBL_OWNER.DBL_RULES**

P	RULE_ID	RAW (16)
UF	TEN_ID	RAW (16)
U	RULE_NAME	VARCHAR2 (100)
U	RULE_NAME_LOWER	VARCHAR2 (100)
U	RULE_TITLE	VARCHAR2 (200)
F	SEV_ID	RAW (16)
F	SEVL_ID	RAW (16)
F	PROF_ID	RAW (16)
F	SQA_ID	RAW (16)
F	IFIX_ID	RAW (16)
	RULE_REASON	CLOB
	RULE_HAS_CHECK	BOOLEAN
	RULE_HAS_QUICK_FIX	BOOLEAN
	RULE_CHECK_IS_FREE	BOOLEAN

DBL_RULES_TENANTS_FK

DBL_RULE_PARAMETERS_RULES_FK

...TWARE_QUALITY_AREAS_FK

DBL_RULE_TAGS_RULES_FK

DBL_RULES_ISSUE_FIXING_TIMES_FK

DBL_DBMS_SCOPES_RULES_FK

DBL_REFERENCES_RULES_FK

DBL_EXAMPLE_SETS_RULES_FK

**C tags
DBL_OWNER.DBL_RULE_TAGS**

P	RTAG_ID	RAW (16)
UF	RULE_ID	RAW (16)
UF	TAG_ID	RAW (16)

**C references
DBL_OWNER.DBL_REFERENCES**

P	REF_ID	RAW (16)
UF	RULE_ID	RAW (16)
UF	LT_ID	RAW (16)
U	REF_NAME	VARCHAR2 (100)
U	REF_NAME_LOWER	VARCHAR2 (100)
	REF_URL	VARCHAR2 (256)
	REF_EXPLANATION	CLOB

**C dbms_scopes
DBL_OWNER.DBL_DBMS_SCOPES**

P	RDB_ID	RAW (16)
UF	RULE_ID	RAW (16)
UF	DBMS_ID	RAW (16)
	RDB_MIN_VERSION	VARCHAR2 (60)
	RDB_MAX_VERSION	VARCHAR2 (60)

**C rule_parameters
DBL_OWNER.DBL_RULE_PARAMETERS**

P	RPARAM_ID	RAW (16)
UF	RULE_ID	RAW (16)
UF	PARAM_ID	RAW (16)

DBL_RULE_TAGS_TAGS_FK

P DBL_OWNER.DBL_TAGS

P	TAG_ID	RAW (16)
UF	TEN_ID	RAW (16)
U	TAG_NAME	VARCHAR2 (100)
U	TAG_NAME_LOWER	VARCHAR2 (100)
	TAG_DESCRIPTION	VARCHAR2 (200)

DBL_REFERENCES_LINK_TYPES_FK

P DBL_OWNER.DBL_LINK_TYPES

P	LT_ID	RAW (16)
U	LT_NAME	VARCHAR2 (100)
U	LT_NAME_LOWER	VARCHAR2 (100)
	LT_DESCRIPTION	VARCHAR2 (200)

DBL_DBMS_SCOPES_DBMS_FK

DBL_TESTS_DBMS_SCOPES_FK

P DBL_OWNER.DBL_DBMS

P	DBMS_ID	RAW (16)
U	DBMS_NAME	VARCHAR2 (100)
U	DBMS_NAME_LOWER	VARCHAR2 (100)
	DBMS_DESCRIPTION	VARCHAR2 (200)

**C tests
DBL_OWNER.DBL_TESTS**

P	TEST_ID	RAW (16)
UF	RDB_ID	RAW (16)
U	TEST_NAME	VARCHAR2 (100)
U	TEST_NAME_LOWER	VARCHAR2 (100)
	TEST_SQL_QUERY	CLOB

DBL_RULE_PARAMETERS_PARAMETERS_FK

P DBL_OWNER.DBL_PARAMETERS

P	PARAM_ID	RAW (16)
UF	TEN_ID	RAW (16)
U	PARAM_NAME	VARCHAR2 (100)
U	PARAM_NAME_LOWER	VARCHAR2 (100)
	PARAM_DESCRIPTION	VARCHAR2 (200)
	PARAM_VALUE	CLOB

P DBL_

P	ES_ID	
UF	TEN_ID	
UF	DBMS_ID	
U	ES_NAME	
U	ES_NAME_	
	ES_URL	

DBL_PARAMETERS_TENANTS_FK

P DBL_OWNER.DBL_TENANTS

P	TEN_ID	RAW (16)
U	TEN_NAME	VARCHAR2 (100)
U	TEN_NAME_LOWER	VARCHAR2 (100)
	TEN_SESSION_TTL	NUMBER (10)
	TEN_IS_CORE	BOOLEAN

#1 Size

Size Matters

JSON Datatype Size Limits

32 MB

33'554'432 Bytes

OSON

(Oracle's native binary JSON format)

JSON Size - Examples

```sql
with j (variant, json_string) as (values
      ('rows_as_objects', '
        {
          "emps": [ { "empno": 7788, "name": "Scott", "sal": 3000 },
                    { "empno": 7839, "name": "King", "sal": 5000 },
                    { "empno": 7876, "name": "Adams", "sal": 1100 } ]
        }
      '),
      ('rows_as_arrays', '
        {
          "emps": [ [ 7788, "Scott", 3000 ],
                    [ 7839, "King", 5000 ],
                    [ 7876, "Adams", 1100 ] ]
        }
      ')
    )
select variant,
       lengthb(json_string) as str_len,
       lengthb(replace(json_string, ' ', null)) as str_trimmed_len,
       lengthb(json(json_string)) as oson_len,
       lengthb(json(replace(json_string, ' ', null))) as oson_trimmed_len
 from j;
```

VARIANT	STR_LEN	STR_TRIMMED_LEN	OSON_LEN	OSON_TRIMMED_LEN
rows_as_objects	242	138	125	125
rows_as_arrays	170	75	96	96

16

OSON Storage Representation

```
Variant           Offset Hex Bytes                                       ASCII Chars
───────────────   ─────  ──────────────────────────────────────────     ──────────────
rows_as_objects   000000 FF 4A 5A 01 21 06 04 00 14 00 50 00 01 42 6F BA  .JZ.!.....P..Bo.
                  000010 E6 00 00 00 10 00 05 00 0B 04 65 6D 70 73 05 65  ..........emps.e
                  000020 6D 70 6E 6F 04 6E 61 6D 65 03 73 61 6C 84 01 01  mpno.name.sal...
                  000030 00 05 C0 03 00 0D 00 25 00 3A 86 03 03 04 02 00  .......%.:......
                  000040 18 00 1C 00 22 22 C2 4E 59 05 53 63 6F 74 74 21  ...."".NY.Scott!
                  000050 C2 1F 9C 00 0D 00 2E 00 32 00 37 22 C2 4F 28 04  ........2.7".O(.
                  000060 4B 69 6E 67 21 C2 33 9C 00 0D 00 43 00 47 00 4D  King!.3....C.G.M
                  000070 22 C2 4F 4D 05 41 64 61 6D 73 21 C2 0C           ".OM.Adams!..

rows_as_arrays    000000 FF 4A 5A 01 21 06 01 00 05 00 4B 00 00 42 00 00  .JZ.!.....K..B..
                  000010 04 65 6D 70 73 84 01 01 00 05 C0 03 00 0D 00 22  .emps.........."
                  000020 00 36 C0 03 00 15 00 19 00 1F 22 C2 4E 59 05 53  .6........".NY.S
                  000030 63 6F 74 74 21 C2 1F C0 03 00 2A 00 2E 00 33 22  cott!.....*...3"
                  000040 C2 4F 28 04 4B 69 6E 67 21 C2 33 C0 03 00 3E 00  .O(.King!.3...>.
                  000050 42 00 48 22 C2 4F 4D 05 41 64 61 6D 73 21 C2 0C  B.H".OM.Adams!..
```

see node-oracledb OSON decoder: https://github.com/oracle/node-oracledb/blob/v6.10.0/lib/impl/datahandlers/oson.js#L280

17

OSON or JSON Datatype Limitation?

OSON BLOB

```
create table i (
    data blob check (data is json format oson)
);

exec dbms_random.seed(42);

insert into i (data)
select json_arrayagg(
        json{
          '_id'  : rownum,
          'name' : initcap(dbms_random
                            .string('A', 30)),
          'sal'  : trunc(dbms_random
                         .value(1000, 5000), 2)
        } returning blob
      )
  from dual connect by level <= 750000;

ORA-40604: exceeded maximum size for a JSON data type object
JZN-00092: JSON too large for binary encoding
```

Conventional BLOB

```
create table i (
    data blob check (data is json)
);

exec dbms_random.seed(42);

insert into i (data)
select json_arrayagg(
        json{
          '_id'  : rownum,
          'name' : initcap(dbms_random
                            .string('A', 30)),
          'sal'  : trunc(dbms_random
                         .value(1000, 5000), 2)
        } returning blob
      )
  from dual connect by level <= 750000;

select length(data) from i;

LENGTH(DATA)
------------
    51548929
```

18

Solutions for ORA-40604/JZN-00092?

Process JSON Chunks < 32 MB

```
create json relational duality view dv as
t @insert @update @delete
{
    _id: empno,
    name: name,
    sal: sal
};

insert into dv (data)
select jt.emp
  from i
 cross apply json_table(
         i.data,
         '$[*]' columns (
             emp json path '$'
         )
       ) jt;
```

Avoid OSON and DVs

```
insert into t (empno, name, sal)
select jt.empno, jt.name, jt.sal
  from i
 cross apply json_table(
         i.data,
         '$[*]' columns (
             empno number               path '$._id',
             name  varchar2(30 char) path '$.name',
             sal   number(7,2)        path '$.sal'
         )
       ) jt;
```

Faster,
Max. 32 TB
with 8 KB Blocks

#2 Surrogate Keys
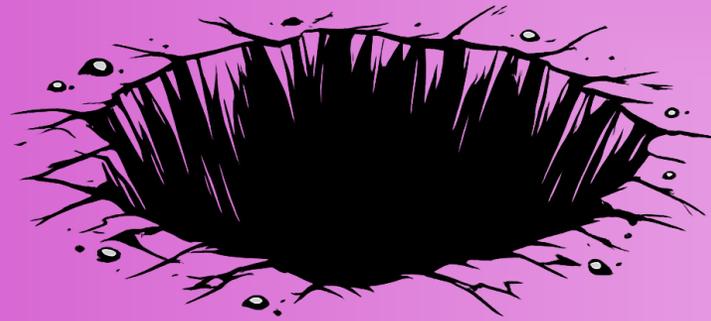
Surrogate Keys – Overview

Typical Characteristics

- System generated (Sequence, UUID)
- Not derived from other business values
- Never reused
- Not manipulable by the user or system
- No semantic meaning
- Exposed to the user or application for reference purposes only
- Not shared across systems

Values

- Single-column PKs and FKs (no composite and compound keys)
- Simplifies access and joins
- No PK and FK updates
- Simplifies BK updates
- Might improve join performance

Model

```
create table d (
  dept_id    integer generated always as identity not null,
  dept_name varchar2(30 char) not null,
  dept_desc varchar2(100 char),
  constraint d_pk primary key (dept_id),
  constraint d_uk unique (dept_name)
);

create table e (
  emp_id    integer generated always as identity not null,
  emp_no    integer not null,
  emp_name varchar2(30 char) not null,
  dept_id  integer not null,
  constraint e_pk primary key (emp_id),
  constraint e_uk1 unique (emp_no),
  constraint e_uk2 unique (emp_name),
  constraint e_d_fk foreign key (dept_id) references d
);
```

Duality View (PK)

```
d @insert @update @delete
{
    _id: dept_id,
    dept_name,
    dept_desc
    emps: e @insert @update @delete
    {
        emp_id,
        emp_no,
        emp_name
    }
};
```

Ensures that sequences remain unique

Model enforces server- side surrogate keys

22

JSON for _id 2

```
{
  "_id" : 2,
  "_metadata" :
  {
    "etag" : "DFA31E21B395F4196E69AE43F8E692B9",
    "asof" : "00000000006AFCE2"
  },
  "dept_name" : "Research",
  "dept_desc" : "Chicago",
  "emps" :
  [
    {
      "emp_id" : 2,
      "emp_no" : 7566,
      "emp_name" : "Jones"
    },
    {
      "emp_id" : 3,
      "emp_no" : 7788,
      "emp_name" : "Scott"
    }
  ]
}
```

Good for original system

Not optimal for sharing across systems

23

Duality View Based on Business Keys?

```
                    Duality View (BK)
create json relational duality view dv_bk as
d @insert @update @delete
{
  _id: dept_name,
  dept_desc,
  emps: e @insert @update @delete
  {
    emp_no,
    emp_name
  }
};


ORA-42550: Cannot create updatable JSON-relational
duality view 'DV_BK': linking column 'DEPT_ID' of
table 'E' should be selected.
```

- no FK columns based on BK
- Missing relationship between D and E
- Read-only DV is possible
- Updatable DV on BK would require redundant data
- Updatable DV on BKs are only feasible if BKs are used exclusively for PKs and FKs

Solution – 1/3) Update DV

Duality View (PK, BK)

```
create or replace json relational duality view dv as
d @insert @update @delete
{
    _id: dept_id
    dept_name
    dept_desc
    emps: e @insert @update @delete
    {
        emp_no
        emp_name
    }
};
```

- Remove PK emp_id
- Works when not used as FK
- Object identified via BK (UK)
- But we have two UKs
  - emp_no
  - emp_name

Solution – 2/3) Remove PK, Metadata

Query

```
select json_serialize(
        json_transform(
            data,
            remove '$._id',
            remove '$._metadata'
        ) pretty
    ) as export_json
  from dv dv where dv.data.dept_name = 'Research';
```

Result

```
{
  "dept_name" : "Research",
  "dept_desc" : "Chicago",
  "emps" :
  [
    {
      "emp_no" : 7566,
      "emp_name" : "Jones"
    },
    {
      "emp_no" : 7788,
      "emp_name" : "Scott"
    }
  ]
}
```

Solution – 3/3) Add PK on Import

Import Procedure

```sql
create or replace procedure import(in_json in json) is
    cursor c_dept is
        select dept_id
          from d
         where dept_name = json_value(in_json, '$.dept_name');
    r_dept c_dept%rowtype;
    cursor c_transform is
        select json_transform(
                 in_json,
                 set '$._id' = r_dept.dept_id create on missing
               ) as data;
    r_transform c_transform%rowtype;
begin
    open c_dept;
    fetch c_dept into r_dept;
    close c_dept;
    open c_transform;
    fetch c_transform into r_transform;
    close c_transform;
    if r_dept.dept_id is not null then
        update dv dv
           set data = r_transform.data
         where dv.data."_id" = r_dept.dept_id;
    else
        insert into dv (data) values (r_transform.data);
    end if;
end import;
```

Call

```sql
begin
    import(json('
      {
        "dept_name" : "Research",
        "dept_desc" : "Chicago2",
        "emps" :
        [
          {
            "emp_no" : 7566,
            "emp_name" : "Jones2"
          },
          {
            "emp_no" : 7788,
            "emp_name" : "Scott2"
          }
        ]
      }'));
end;


select * from e natural join d;

DEPT_ID EMP_ID EMP_NO EMP_NA DEPT_NAME   DEPT_DESC
------- ------ ------ ------ ---------- ----------
      1      1   7839 King   Accounting New York
      2      4   7566 Jones2 Research   Chicago2
      2      5   7788 Scott2 Research   Chicago2
```

Delete/Insert for each emp due to UK update

First UK by property name is used

27

Solution Approach in dbLinter Repo

Model

- Updateable DVs based on PK/FK
- All PK based on surrogate keys with
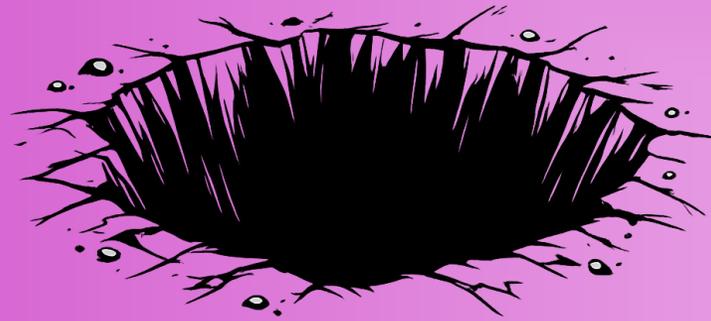 `default on null for insert only sys_guid()`

Why not use "generated always" identity columns?
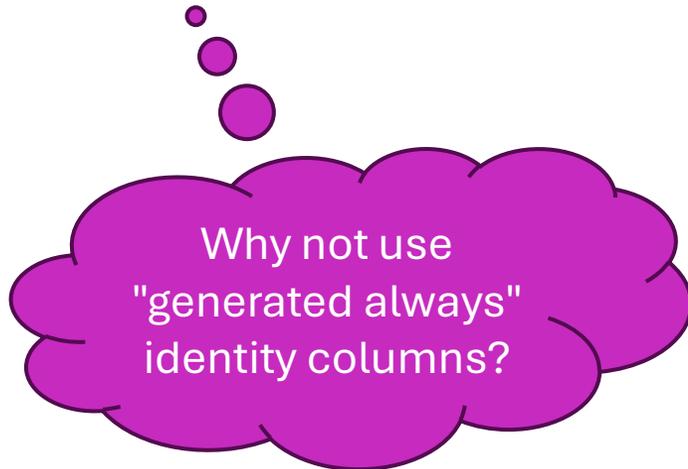
Export

- Remove _id node
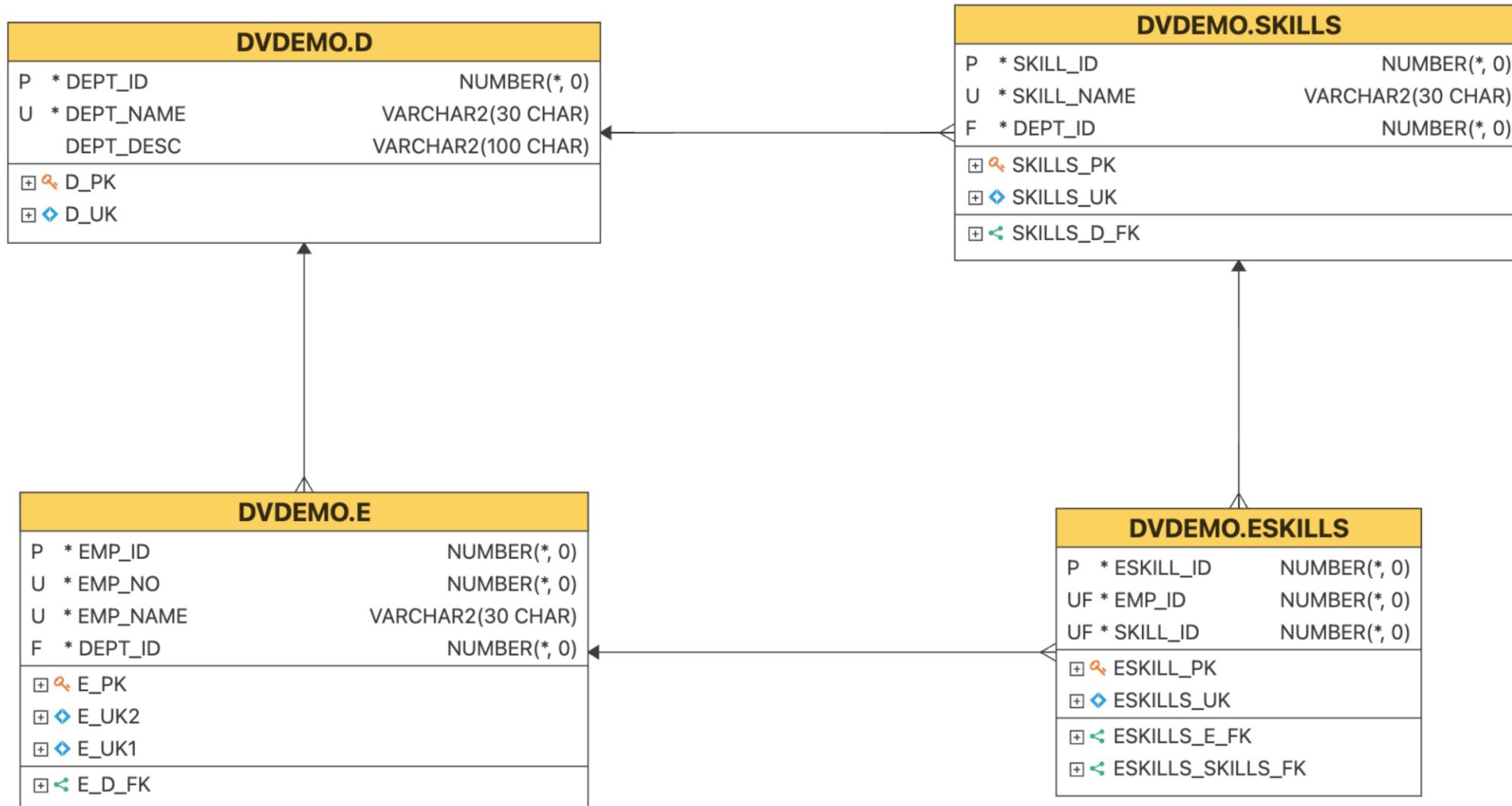- Remove _metadata node
- Remove all surrogate keys

Import

- Populate surrogate keys via BK (an explicitly chosen UK)
- Insert via DV (no _id)
- Update via DV (existing _id)

#3 Referential Integrity

Model – Overview

Model – Sequence, Tables

```
create sequence ids;

create table d (
  dept_id   integer default on null
                         for insert only ids.nextval,
  dept_name varchar2(30 char) not null,
  dept_desc varchar2(100 char),
  constraint d_pk primary key (dept_id),
  constraint d_uk unique (dept_name)
);

create table skills (
  skill_id   integer default on null
                          for insert only ids.nextval,
  skill_name varchar2(30 char) not null,
  dept_id    integer not null,
  constraint skills_pk primary key (skill_id),
  constraint skills_uk unique (skill_name),
  constraint skills_d_fk foreign key (dept_id)
                                        references d
);
```

```
create table e (
  emp_id    integer default on null
                             for insert only ids.nextval,
  emp_no    integer not null,
  emp_name varchar2(30 char) not null,
  dept_id   integer not null,
  constraint e_pk primary key (emp_id),
  constraint e_uk1 unique (emp_no),
  constraint e_uk2 unique (emp_name),
  constraint e_d_fk foreign key (dept_id) references d
);

create table eskills (
  eskill_id integer default on null
                             for insert only ids.nextval,
  emp_id    integer not null,
  skill_id  integer not null,
  constraint eskill_pk primary key (eskill_id),
  constraint eskills_uk unique (emp_id, skill_id),
  constraint eskills_e_fk foreign key (emp_id)
                                        references e,
  constraint eskills_skills_fk foreign key (skill_id)
                                        references skills
);
```

Duality View

```
create or replace json relational duality view dv as
d @insert @update @delete
{
    _id: dept_id
    dept_name
    dept_desc
    skills: skills @insert @update @delete
    {
        skill_id
        skill_name

    }
    emps: e @insert @update @delete
    {
        emp_id
        emp_no
        emp_name
        skills: eskills @insert @update @delete
        {
            eskill_id
            skills @unnest
            {
                skill_id
                skill_name
                d @unnest
                {
                    dept_id @nocheck
                    dept_name @nocheck
                }
            }
        }
    }
};
```

Insert into DV

```
insert into dv (data) values (json('{
    "_id" : 1,
    "dept_name" : "HR",
    "dept_desc" : "Human Resources",
    "skills" :
    [
        {
            "skill_id" : 2,
            "skill_name" : "SQL"
        },
        {
            "skill_id" : 3,
            "skill_name" : "AI"
        }
    ],
    "emps" :
    [
        {
            "emp_id" : 4,
            "emp_no" : 1234,
            "emp_name" : "Luma",
            "skills" :
            [
                {
                    "eskill_id" : 5,
                    "skill_id" : 3,
                    "skill_name" : "AI",
                    "dept_id" : 1,
                    "dept_name" : "HR"
                }
            ]
        }
    ]
}'));
```

```
ORA-42692: Cannot insert into JSON Relational Duality View
'DVDEMO'.'DV': Error while inserting into table 'ESKILLS'
ORA-02291: integrity constraint (DVDEMO.ESKILLS_SKILLS_FK)
violated - parent key not found
```

Solution – Check Integrity At Commit

```
alter table eskills drop constraint eskills_skills_fk;
alter table skills drop constraint skills_pk;

alter table skills
    add constraints skills_pk
        primary key (skill_id) rely;

alter table eskills
    add constraint eskills_skills_fk
        foreign key (skill_id) references skills
        deferrable initially immediate rely;

set constraints all deferred;
insert into dv (data) values (...);
commit;
```

"deferrable" changes default behaviour of RELY!

May lead to different execution plans!

We defer the check for DV DMLs only

33

#4 Composition vs. Aggregation

Composition

```
                  Dept with Emps
{
  "_id" : 2,
  "dept_name" : "Research",
  "dept_desc" : "Chicago",
  "emps" :
  [
    {
      "emp_id" : 4,
      "emp_no" : 7566,
      "emp_name" : "Jones"
    },
    {
      "emp_id" : 5,
      "emp_no" : 7788,
      "emp_name" : "Scott"
    }
  ]
}
```

- Single document structure
- Linked life cycle
- Limited update flexibility

Aggregation

```
●  ●  ●                Departments

{
  "_id" : 2,
  "dept_name" : "Research",
  "dept_desc" : "Chicago"
}
```

```
●  ●  ●                Employees

{
  "_id" : 4,
  "emp_no" : 7566,
  "emp_name" : "Jones",
  "dept_id" : 2,
  "dept_name" : "Research"
}

{
  "_id" : 5,
  "emp_no" : 7788,
  "emp_name" : "Scott",
  "dept_id" : 2,
  "dept_name" : "Research"
}
```

- Multiple document structure
- Independent life cycle
- Update flexibility

36

One Model – Multiple Representations

Composition

```
create json duality view dv as
d @insert @update @delete
{
    _id: dept_id,
    dept_name,
    dept_desc
    emps: e @insert @update @delete
    {
        emp_id,
        emp_no,
        emp_name
    }
};
```

Aggregation

```
create json duality view d_dv as
d @insert @update @delete
{
    _id: dept_id
    dept_name
    dept_desc
};

create json duality view e_dv as
e @insert @update @delete
{
    _id: emp_id
    emp_no
    emp_name
    d @unnest
    {
        dept_id
        dept_name
    }
};
```

Composition Dept with Emps

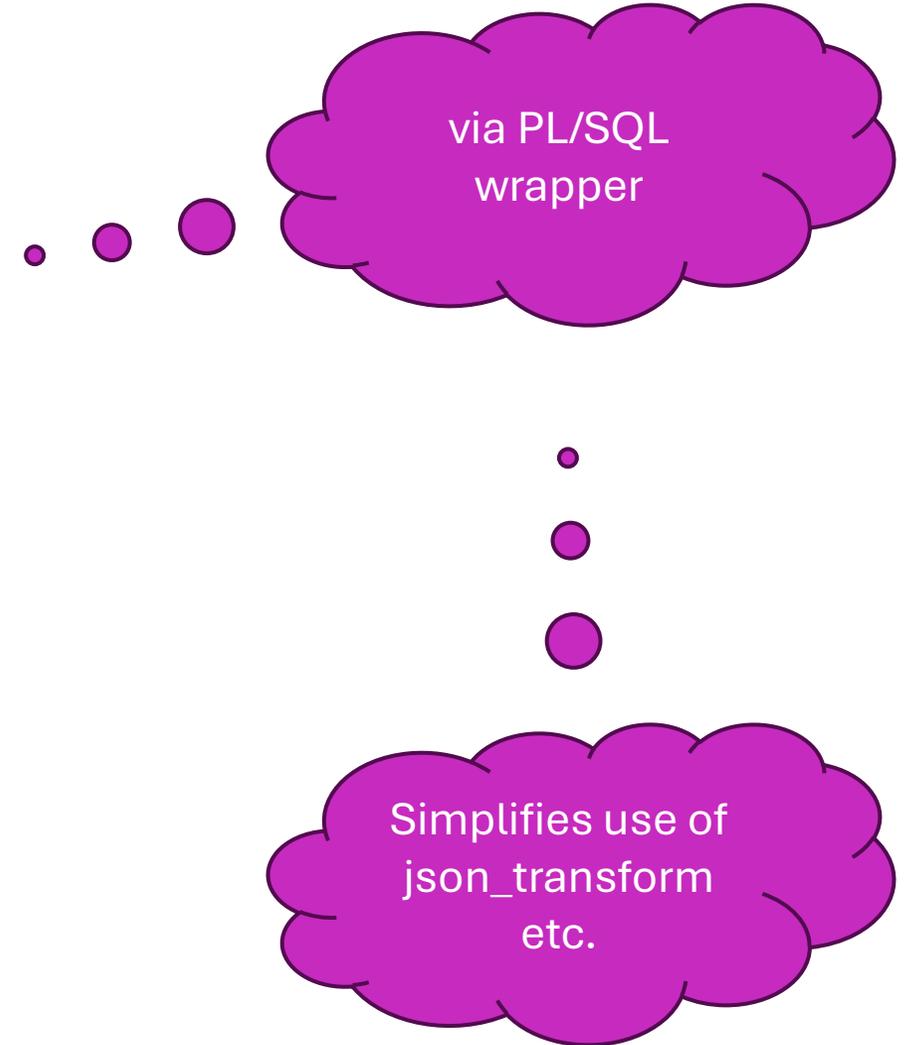| Operation	emps Array	emps @insert @update @delete	emps @noinsert @noupdate @nodelete
Insert	emps: []	Deletes emps	Keeps emps (deferred or disabled constraints)
Update	emps: []	Deletes emps	Keeps emps
Update	emps: [{Jones}]	Deletes emps except Jones Insert or Updates Jones	ORA-40938: Cannot delete from table 'E' in JSON Relational Duality View 'DV': Missing DELETE annotation or NODELETE annotation specified.
Delete	-	Deletes emps	Succeeds for depts without emps ORA-40938: Cannot delete from table 'E' in JSON Relational Duality View 'DV': Missing DELETE annotation or NODELETE annotation specified.

other annotation combinations?

rarely relevant

also due to technical delete/insert operations instead of a single update

JSON in dbLinter Repo

- Duality Views for Export & Import
    - Tenants
    - Users

- Duality Views for Export only
    - Rules (for https://dblinter-rules.united-codes.com/)
    - Logs

- SQL/JSON for read-only access
    - Open Client Session

via PL/SQL wrapper

Simplifies use of json_transform etc.

Core Messages

JSON Documents in Oracle AI Database

- Keep JSON documents in duality views small

- Use SQL/JSON for large JSON documents

- Consider use case-specific (duality) views / JSON documents

- Defer constraint checks for embedded relationships in updatable duality views

JSON Documents Shared Across Systems

- Remove surrogate keys from source system

- Add surrogate keys from target system based on business keys

- Never use "generated always" identity columns for embedded relationships

Thank you!

GRISSELBAV
simple & accurate IT solutions.