



# KONFERENZ + AUSSTELLUNG

**Oracle 23ai:  
Domains & Annotations  
Deep Dive**

Dienstag, 19. November 2024 (11:00 - 11:45)



**Robert Marz**  
**DATABEE**  
Die IT-Architekten

[ANWENDERKONFERENZ.DOAG.ORG](https://anwenderkonferenz.doag.org)



Robert Marz – Independent Consultant

## Primary Role

Senior Technical Architect  
with database centric view of the world

~~SOAG  
(German Oracle User Group)~~

~~Active Member of Database Community  
Responsible for Cloud Topics~~



**DATABEE**  
Die IT-Architekten



**Databees.**

**.SYM<sup>L2</sup>**



**Oracle ACE  
Pro**



@RobbieDabee



<https://robbie.databee.org>



robert.marz@dabee.org



# Robert Marz – Independent Consultant

## Primary Role

Senior Technical Architect  
with database centric view of the world

## ora2know

The German Oracle Database Community.  
Database first. Community first. [ora2know.de](http://ora2know.de)  
Member of the Board



**DATABEE**  
Die IT-Architekten



**Databees.**



**ora2know**  
The German Oracle Database Community

**SYM<sup>L2</sup>**



@robbie.databee.org



@RobbieDatabee



<https://robbie.databee.org>



robert.marz@databee.org



**Oracle ACE  
Pro**



**Eine starke Community für Oracle Datenbank Benutzer**

**Ziele: Wissensvermittlung für Entwickler und DBAs**

**Angebote: MeetUps, Podcasts, Live Events**

**Projekte: Veranstaltung zum Kennenlernen**

**Kontakt: [info@ora2know.de](mailto:info@ora2know.de)**



**ora2know**

The German Oracle Database Community



# The Oracle ACE Program

400+ technical experts helping peers globally



- The Oracle ACE Program recognizes and rewards community members for their technical and community contributions to the Oracle community
- 3 membership levels: Director, Pro, and Associate
- Nominate yourself or a colleague at [ace.oracle.com/nominate](https://ace.oracle.com/nominate)
- Learn more at [ace.oracle.com](https://ace.oracle.com)



# SYMPOSIUM<sup>42</sup>

Created by the community, to support the community

Sharing of reliable knowledge

Supporting the various user groups and individuals



@sym\_42



<https://sym42.org/>

# Oracle 23ai





# Oracle 23ai Feature Overview

The diagram features a central green box with the text "Oracle Database 23ai Bring AI to your data". Surrounding this central box are 18 blue feature tiles, each with an icon and a title. The features are arranged in a grid-like fashion around the center.

- Data Use Case Domains**: Icon showing a circular diagram with labels like "ACCOUNT", "ZIP", "EMAIL", "ORDER", "ID", "SSN".
- Boolean Datatype**: Text only.
- Real-time SQL Plan Management**: Icon showing a flow from "SQL" to a red "X" and a green checkmark.
- JSON Schema**: Icon showing a document with a red tab.
- Readable PDB Standby**: Icon showing a person, a database cylinder, and a cylinder labeled "R/O".
- Property Graphs**: Icon showing a network graph with red nodes.
- Microservice Sagas**: Icon showing two gears and two database cylinders.
- JSON / Relational Duality**: Icon showing a "{JSON}" box and a table.
- AI Vector Search**: Icon showing a brain with a gear and circuitry.
- True Cache**: Icon showing three "CACHE" boxes connected to a "23c" database cylinder.
- SQL Firewall**: Icon showing a padlock.
- Priority Transactions**: Text only.
- JS Stored Procedures**: Icon showing the "JS" logo.
- Developer Role**: Icon showing a person with glasses.
- Shrink Tablespace**: Text only.
- Schema Privileges**: Text only.
- Globally Distributed Database**: Icon showing a globe with location pins and three database cylinders below.
- Rolling Patching**: Icon showing a database cylinder with a circular arrow and two green checkmarks.





# Annotations – Modern Comments

**Comments** Separate Objects  
single text line

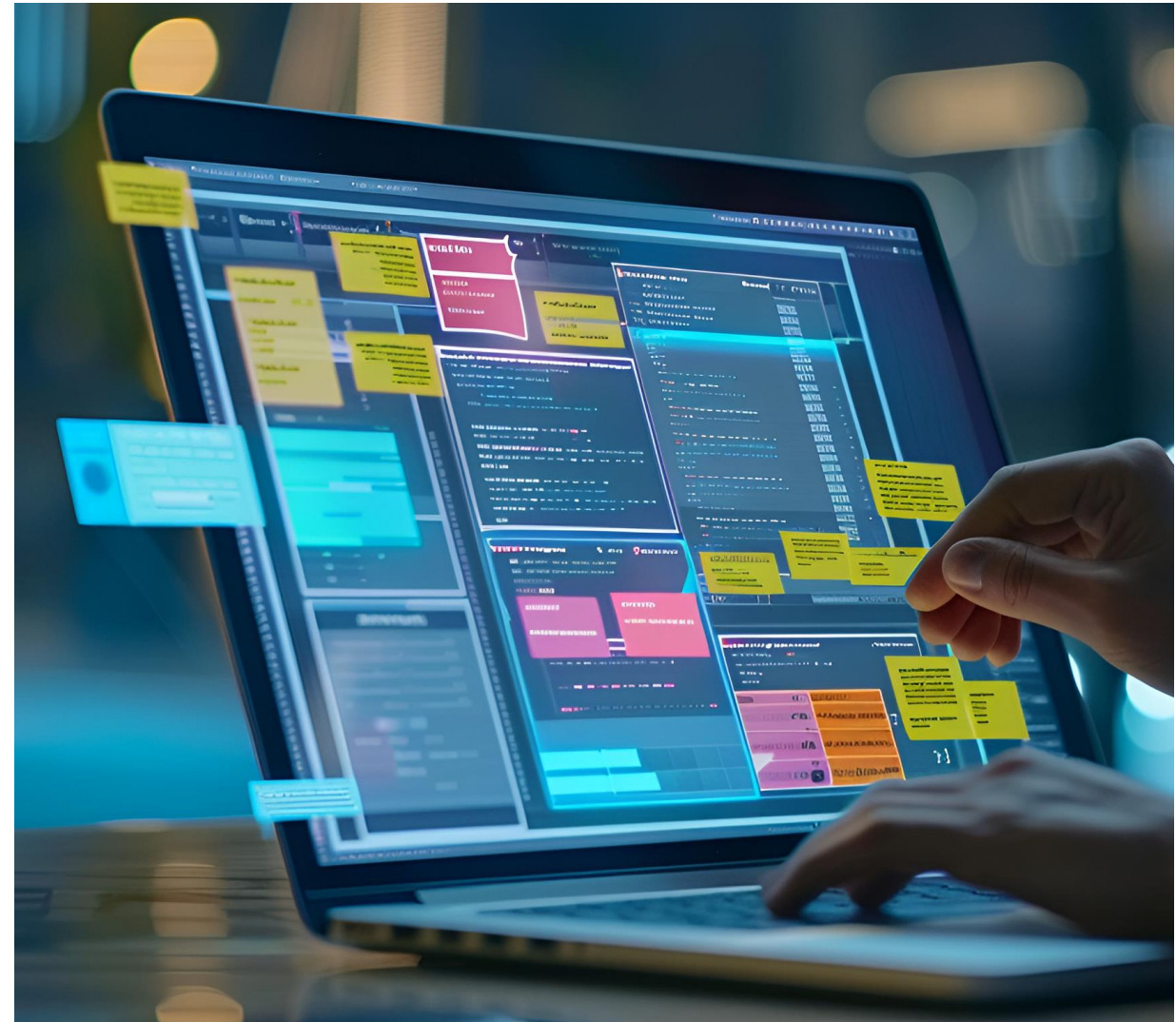
- per table
- per column

Apply to

- Tables
- Views
- Materialized Views

**Annotations** Tied to their Object  
Many Key-Value-Pairs  
Annotatable Objects

- Same as Comments
- Indexes
- (more to come)





# Annotations: Key-Value Pairs

## Keys and Values

- free Text

## Keys

- mandatory
- 1-1024 characters

## Values

- optional
- 0-4000 characters
- JSON possible

```
create table job_assignments (  
  emp_id      raw(16) not null  
             annotations (SurrogateKey,  
                           PK), -- key only annotations  
  
  job_id      raw(16) not null  
             annotations (SurrogateKey, -- Indicates that these columns  
                           PK), -- is part of the primary key. Remember,  
                           -- annotations are free form.  
  
  start_ts    timestamp with time zone  
             annotations (content 'actual start of work',  
                           UI_Display  
                           [{"format": "long"},  
                            {"tz": "utc"},  
                            {"label": "Work Start"}]),  
  
  changed_by  varchar2(100)  
             annotations (UI_Display [{"visibility": "hidden"}],  
                           Classification 'change tracking'),  
  
  changed     timestamp with time zone  
             annotations (UI_Display [{"visibility": "hidden"}],  
                           Classification 'change tracking')  
)  
annotations (relation ["emps", "jobs"],  
             content 'Assignments of employees to jobs');
```



# Annotations: Syntax

## Annotations

- cannot be created standalone

## create object

- annotations-clause

## alter object

- add
- drop
- replace

## modifier

- if [not] exists

```
alter table job_assignments
  modify start_ts
  annotations (
    drop if exists "CONTENT",
    add if not exists missing_annotation,
    replace ui_display
      '[{"format": "short"},
        {"tz": "local"},
        {"label": "Work Start"}]');
```



# Annotations: Metadata Views

[USER|ALL|DBA]\_

ANNOTATIONS

Annotation Names

ANNOTATIONS\_USAGE

Most Details

ANNOTATION\_VALUES

Only Values

```
-- Metadata query
select column_name,
       annotation_name,
       annotation_value
from   user_annotations_usage
where  object_name = 'JOB_ASSIGNMENTS'
order  by column_name nulls first ,annotation_name, annotation_value;
```

COLUMN_NAME	ANNOTATION_NAME	ANNOTATION_VALUE
	CONTENT RELATION	Assignments of employees to jobs ["emps", "jobs"]
CHANGED	CLASSIFICATION	change tracking
CHANGED	UI_DISPLAY	[{"visibility": "hidden"}]
CHANGED_BY	CLASSIFICATION	change tracking
CHANGED_BY	UI_DISPLAY	[{"visibility": "hidden"}]
EMP_ID	PK	
EMP_ID	SURROGATEKEY	
JOB_ID	PK	
JOB_ID	SURROGATEKEY	
START_TS	MISSING_ANNOTATION	
START_TS	UI_DISPLAY	[{"format": "short"}, {"tz": "local"}, {"label": "Work Start"}]

A hand holding a magnifying glass over a small cube on a cobblestone surface. The magnifying glass is positioned over a small, grey, textured cube that sits on a surface of dark, irregular cobblestones. The background is blurred, showing more of the cobblestone surface and some indistinct shapes. The lighting is soft, creating a focused effect on the cube.

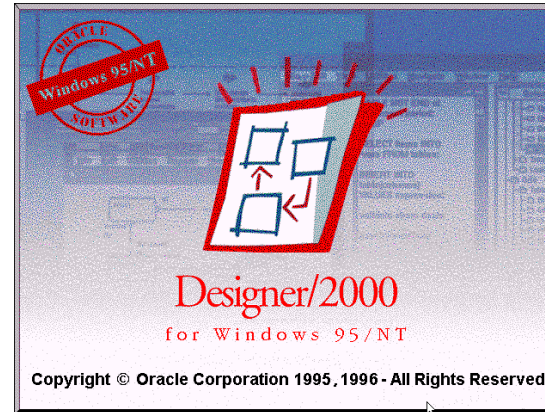
# Data Use Case Domains



# Data Use Case Domains - History

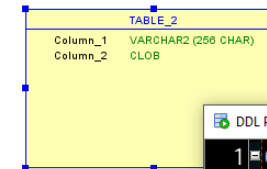
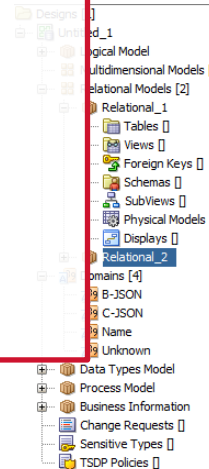
## Domains in Design Tools

- “since ever”
- define custom data types with addons
- keep data type consistent across all tables
- Change all columns at once



## Domains the Database

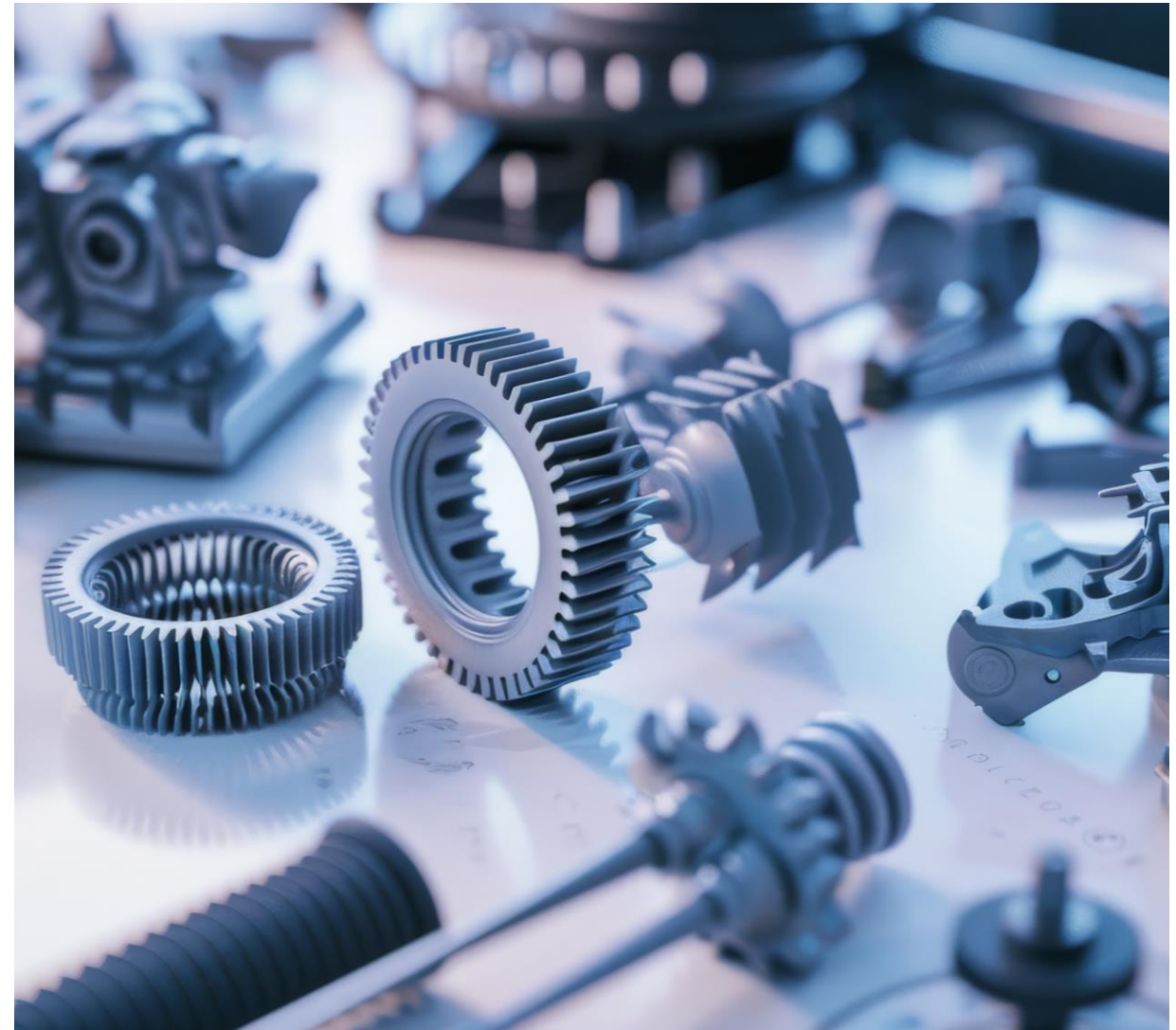
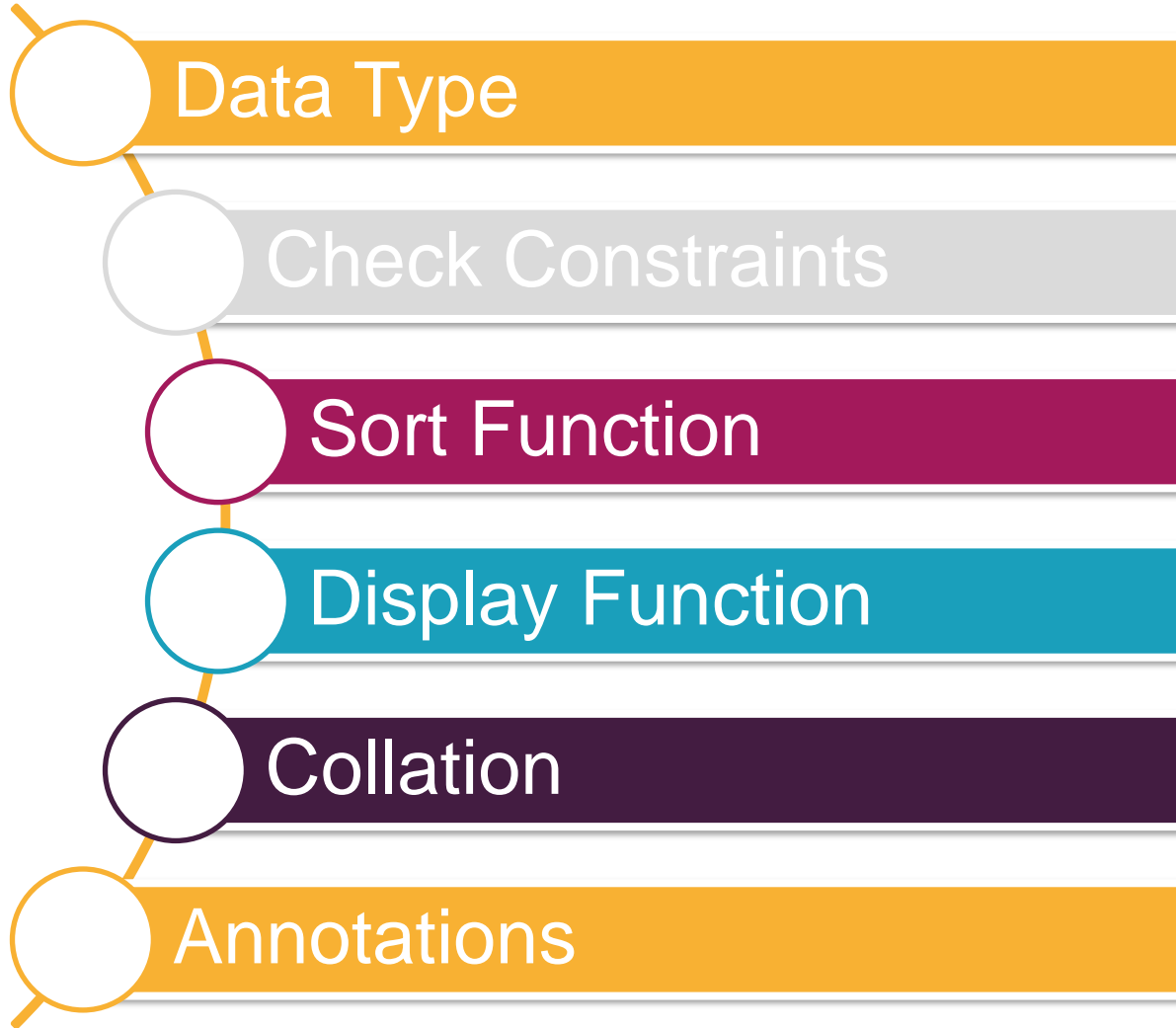
- SQL-Standard:2023
- Oracle 23ai – with enhancements
- PostgreSQL



```
DDL Preview
1 CREATE TABLE table_2 (
2   column_1  VARCHAR2 (256 CHAR),
3   column_2  CLOB
4 );
5
6 ALTER TABLE table_2 ADD CHECK (column_2 IS JSON);
```



# Data Use Case Domains: Components





# Data Use Case Domains: Syntax Example

```
create domain uuid
  as varchar2(36)
  not null
  check (regexp_like(uuid,
    '^([0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12})$'
  ));

create table jobs(job_id uuid);

desc jobs

Name          Null?     Typ
-----
JOB_ID        NOT NULL VARCHAR2(36) DOMAIN UUID

insert into jobs(job_id)
  values ('12345678-1234-1234-1234-123456789abc');

-- 1 row inserted.

insert into jobs(job_id)
  values ('1234');

-- ORA-11534: check constraint (ROBBIE.SYS_C008419)
-- involving column JOB_ID
-- due to domain constraint ROBBIE.SYS_DOMAIN_C0045
-- of domain ROBBIE.UUID violated
```

```
create domain uuid
  as varchar2(72);

create table j1(job_id varchar2(72));
create table j2(job_id varchar2(36));

alter table j1 modify (job_id) add domain uuid;
-- Table J1 altered.

alter table j2 modify (job_id) add domain uuid;
-- Table J2 altered.

drop domain if exists uuid force;

create domain uuid
  as varchar2(72) strict;

alter table j2 modify (job_id) add domain uuid;
-- ORA-11517: the column data type does not
-- match the domain column
```



# Data Use Case Domains: Validate JSON

```
create table jobs
(job json validate
 '{ "type": "object",
   "properties": { "job_id": { "type": "string"} }
 }');
```

```
create domain json_job as json validate '{
  "type": "object",
  "properties": {
    "job_id": {
      "type": "string",
      regexp: "^[0-9a-f]
    }
  },
  "required": ["job_id"]
}';
```

```
create table jobs2(job json_jo
```

```
select table_name, SEARCH_CONDITION
from user_constraints
where table_name Like 'JOBS%'
and constraint_type = 'C';
```

TABLE_NAME	SEARCH_CONDITION
JOBS	"JOB" IS JSON VALIDATE '{ "type": "object", "properties": {
JOBS2	"JOB" IS JSON FORMAT JSON (LAX VALIDATE '{"type": "object", "properties": {"job_id

```
select *
from user_json_domain_schema_columns;
```

DOMAIN_NAME	COLUMN_NAME	CONSTRAINT_NAME	JSON_SCHEMA
JSON_JOB	JSON_JOB	SYS_DOMAIN_C0052	{"type": "object", "properties": {"job_id": {"type": "s



# Data Use Case Domains: Functions

Part of  
domain

domain\_display(<column>)

domain\_sort(<column>)

Identify  
& Test

domain\_name(<column>)

domain\_check(<domain\_name>, <value>)

domain\_check\_type(<domain\_name>, <value>)

Cast

cast

(<column> as domain <domain\_name>)

```

create domain short_string as varchar2(30);

create domain uuid
as raw(16) default sys_guid() not null
annotations (UI_Display 'UUID', Classification 'uuid')
display lower(
  substr(lpad(rawtohex(uuid),32,'0'), 1, 8)|| '-' ||
  substr(lpad(rawtohex(uuid),32,'0'), 9, 4)|| '-' ||
  substr(lpad(rawtohex(uuid),32,'0'), 13, 4)|| '-' ||
  substr(lpad(rawtohex(uuid),32,'0'), 17, 4)|| '-' ||
  substr(lpad(rawtohex(uuid),32,'0'), 21));

create table j3
(id uuid primary key,
 name short_string);

insert into j3 (name) values ('23ai Domains');

select id, domain_display(id) from j3;

ID                                DOMAIN_DISPLAY(ID)
-----                                -
22BC6158C0EA05E4E0630200590A169C  22bc6158-c0ea-05e4-e063-0200590a169c

with uuids as
(select sys_guid() as id connect by level ≤ 3)
select id,
  domain_display(id),
  domain_check(uuid, id),
  domain_display(cast (id as domain uuid))
from uuids;

ID                                DOMAIN_DISPLAY(ID)  DOMAI  DOMAIN_DISPLAY(CAST(IDASDOMAINUUID))
-----                                -
24845C04908E04C9E0630200590A862F  24845c04-908e-04c9-e063-0200590a862f  true
24845C04908F04C9E0630200590A862F  24845c04-908f-04c9-e063-0200590a862f  true
24845C04909004C9E0630200590A862F  24845c04-9090-04c9-e063-0200590a862f  true

```



# Data Use Case Domains: Data Dictionary Views



[USER|ALL|DBA]\_

---

DOMAINS

---

DOMAIN\_COLS

---

DOMAIN\_CONSTRAINTS

---

JSON\_DOMAIN\_SCHEMA\_COLUMNS

---



# Multi Column Domains



# Multi Column Data Use Case Domains

## Multiple Columns

- can be grouped

## Check constraints

- can reference all domain cols

## Table Columns

- with matching data type must be present

```
drop table if exists addresses;
create domain adr_domain as
( street as varchar2(255)
, zip   as varchar2(10)
, city  as varchar2(255)
, country as varchar2(255)
) check (zip is null or regexp_like(zip, '^d{5}$'));
```

```
create table addresses
( id          number primary key,
  street      varchar2(255),
  zip         varchar2(10),
  city        varchar2(255),
  country     varchar2(255),
  domain adr_domain (street, zip,city,country));
```



# Flexible Domains



# Flexible Data Use Case Domains (1/2)

## Define

multiple (multi column) Domains

## Choose

for every table row  
what Domain to use

## Endless Possibilities

e.g. Addresses:  
• choose check constraints and  
• display\_function  
based on country code

## Killer Feature for JSON Validation

- Choose JSON Schema based on content
- Handle different API-Versions
- .....

```
create domain job_type_a
as -- this is a multi-column domain
(job_details as json validate '{
  "type": "object",
  "properties": {"type_a": {"type": "string"}},
  "required": ["type_a"]
}');

create domain job_type_b
as -- add as many column and constraints as needed
(job_details as json validate '{
  "type": "object",
  "properties": {"type_b": {"type": "string"}},
  "required": ["type_b"]
}');

create FLEXIBLE domain job_types
-- number of columns must match subdomains
(job_details)
choose domain using (job_mode varchar2(2 char)) from
-- options: decode() and case statements
-- case with full comparison expression only
case
  when job_mode = 'A' then job_type_a(job_details)
  when job_mode = 'B' then job_type_b(job_details)
  else job_type_a(job_details) -- default
end;
```



## Flexible Data Use Case Domains (2/2)

```
create table jobs_flex
  (id number primary key,
   job_mode varchar2(2 char),
   job_details JSON);

alter table jobs_flex
  modify (job_details, job_mode) add domain job_types;

insert into jobs_flex (id, job_mode, job_details)
  values (1, 'A', '{"type_a": "A job"}'),
         (2, 'B', '{"type_b": "B job"}');
-- 2 rows inserted.
insert into jobs_flex (id, job_mode, job_details)
  values (3, 'A', '{"type_b": "A job"}');

-- ORA-11534: check constraint (ROBBIE.SYS_C008451)
-- involving columns JOB_MODE, JOB_DETAILS
-- due to domain constraint ROBBIE.SYS_DOMAIN_C0059
-- of domain ROBBIE.JOB_TYPES violated
```



A close-up photograph of a hand interacting with a colorful abacus. The abacus has several horizontal rods with beads in various colors: yellow, green, blue, and purple. The hand is positioned over the green beads, with the index finger touching one of them. The background is slightly blurred, showing other parts of the abacus and some wooden blocks. A semi-transparent orange and red banner is overlaid on the left side of the image, containing the text "Enumeration Domains".

# Enumeration Domains



# Data Use Case Domains – Enum „Exotic Flavor“

## Read Only Table

- Key – Value Pairs
- Keys may have aliases
- Keys are Object Names → Uppercase or Quotes
- Check Constraint
  - Ensures only defined Values can be inserted
- Sort- and Display-Function

## Default Data Type

- number sequence starting with 1
- Starting point can be adjusted

## Custom Data Type

- By assigning values
- First assignment determines data type

## Persistence in Code

- Check Constraints
- Domain Functions
- → Not for large Volumes

```
-- generated number sequence starting with 1 as enum values
create domain job_status_domain
  as enum (created,
          scheduled,
          accepted,
          in_progress,
          paused,
          completed);
```

```
-- Enum Domains can be used as read-only tables
select *
  from job_status_domain;
```

ENUM_NAME	ENUM_VALUE
CREATED	1
SCHEDULED	2
ACCEPTED	3
IN_PROGRESS	4
PAUSED	5
COMPLETED	6

```
6 rows selected.
```



# Data Use Case Domains – Enum Basic Usage

```
create table jobs
(
  id number generated always as identity primary key,
  job_name varchar(255),
  status job_status_domain
);

insert into jobs (job_name, status)
values ('Job a', 1),
       ('Job b', 2),
       ('Job c', job_status_domain.accepted),
       ('Job d', job_status_domain.in_progress),
       ('Job e', job_status_domain.paused),
       ('Job f', job_status_domain.completed)
;

6 rows inserted.
```

```
-- Fails because only 6 values are defined in job_status_domain
insert into jobs (job_name, status)
values ('Job g', 7);
```

```
ORA-11534: check constraint (ROBBIE.SYS_C0024412)
involving column STATUS
due to domain constraint ROBBIE.SYS_DOMAIN_C0052
of domain ROBBIE.JOB_STATUS_DOMAIN violated
```

```
select job_name, status, domain_display(status)
from jobs;
```

<u>JOB_NAME</u>	<u>STATUS</u>	<u>DOMAIN_DISPLAY(STATUS)</u>
Job a	1	CREATED
Job b	2	SCHEDULED
Job c	3	ACCEPTED
Job d	4	IN_PROGRESS
Job e	5	PAUSED
Job f	6	COMPLETED



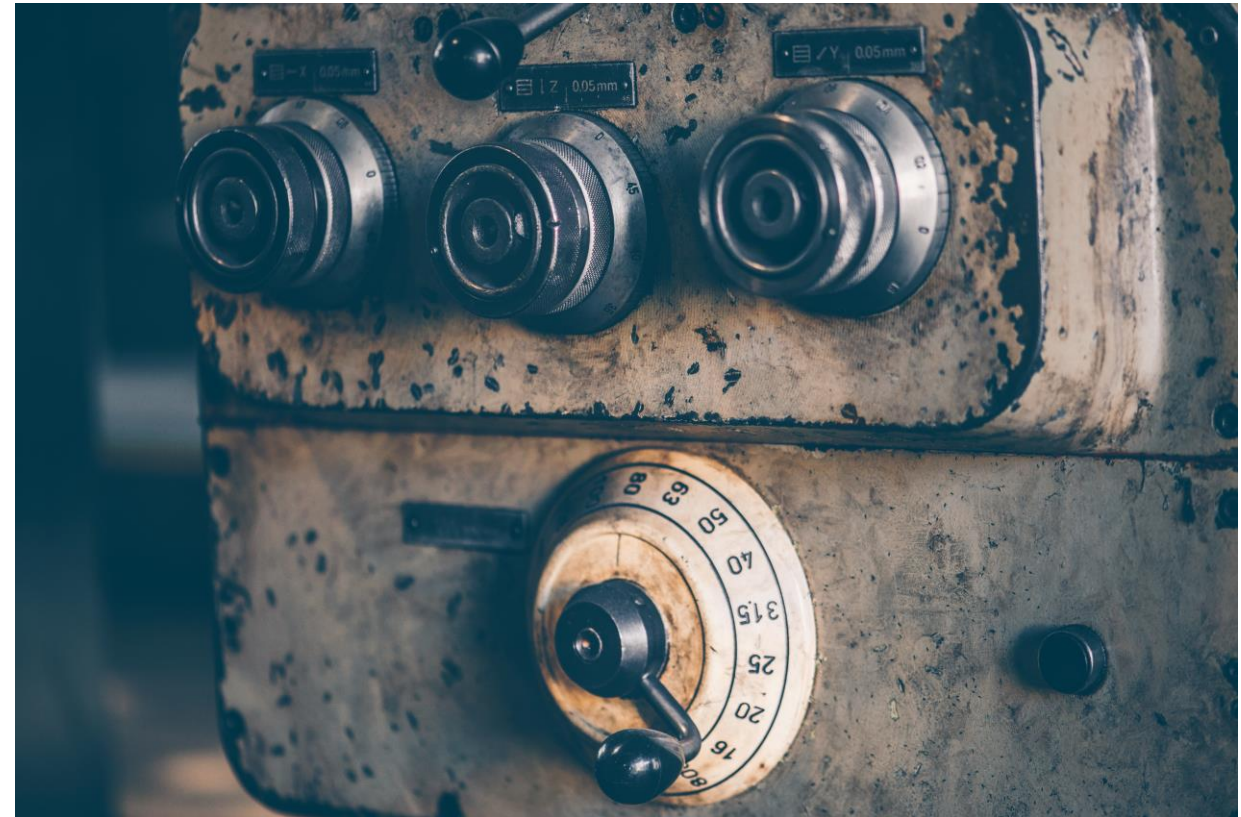
# Data Use Case Domains – Enum Adjust Start Point

```
-- Create basic enumeration domain starting with 0
create domain job_status_domain
  as enum (created = 0,      -- Starting with 0
          scheduled,
          accepted,
          in_progress,
          paused,
          completed);
```

```
select *
  from job_status_domain;
```

<u>ENUM_NAME</u>	<u>ENUM_VALUE</u>
CREATED	0
SCHEDULED	1
ACCEPTED	2
IN_PROGRESS	3
PAUSED	4
COMPLETED	5

6 rows selected.





# Data Use Case Domains – Enum Aliases, Values, Order Function (1)

```
-- Create more complex enumeration domain
-- Use aliases and varchar2 enum values
create domain job_status_domain
  as enum (INS = created = 'Created',
          PLN = planned = 'Scheduled',
          ACC = accepted = 'Accepted',
          PRG = inprogress = 'In Progress',
          HLD = paused = 'On Hold',
          FIN = "done" = 'Completed'
          )
  order case(job_status_domain)
    when 'Created' then 30
    when 'Scheduled' then 40
    when 'Accepted' then 70
    when 'In Progress' then 20
    when 'On Hold' then 80
    when 'Completed' then 10
    else 99
end;
```

```
-- Because of the aliase, we get 12 entries
select *
  from job_status_domain;
```

ENUM_NAME	ENUM_VALUE
INS	Created
CREATED	Created
PLN	Scheduled
PLANNED	Scheduled
ACC	Accepted
ACCEPTED	Accepted
PRG	In Progress
INPROGRESS	In Progress
HLD	On Hold
PAUSED	On Hold
FIN	Completed
done	Completed

12 rows selected.



## Data Use Case Domains – Enum Aliases, Values, Order Function (2)

```
create table jobs
(
  id number generated always as identity primary key,
  job_name varchar(255),
  status job_status_domain
);

insert into jobs (job_name, status)
values ('Job a', 'Created'),
       ('Job b', job_status_domain.planned),
       ('Job c', job_status_domain.acc),
       ('Job d', job_status_domain.prg),
       ('Job e', job_status_domain.hld),
       ('Job f', job_status_domain."done")
;      -- Note the double quotes
```

```
select job_name, status
       , domain_display(status) as disp_f
       , domain_order(status) as order_f
  from jobs
 order by domain_order(status);
```

<u>JOB_NAME</u>	<u>STATUS</u>	<u>DISP_F</u>	<u>ORDER_F</u>
Job f	Completed	FIN	10
Job d	In Progress	PRG	20
Job a	Created	INS	30
Job b	Scheduled	PLN	40
Job c	Accepted	ACC	70
Job e	On Hold	HLD	80

6 rows selected.



# Data Use Case Domains – Check Constraint

```
desc jobs
```

Name	Null?	Type
ID	NOT NULL	NUMBER
JOB_NAME		VARCHAR2(255)
STATUS		VARCHAR2(22) DOMAIN JOB_STATUS_DOMAIN

```
select table_name, constraint_type, constraint_name, SEARCH_CONDITION  
  from user_constraints  
 where table_name = 'JOBS';
```

TABLE_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
JOBS	C	"ID" IS NOT NULL
JOBS	C	("STATUS"='Created' OR "STATUS"='Scheduled' OR "STATUS"='Accepted' OR "STATUS"='In Progress' OR "STATUS"='On Hold' OR "STATUS"='Completed')
JOBS	P	



# Pre-defined Domains



# Pre Defined Data Use Case Domains

There are 109 (23.6) predefined Domains

## Technical

UUID4

IPv4 Address

IPv6 Address

Subnet Mask

CIDR

## Mathematical

Measure

Percent

Ratio

Numbers (Negative, Postive, Non-Zero)

## Communications

Email

Social Media IDs

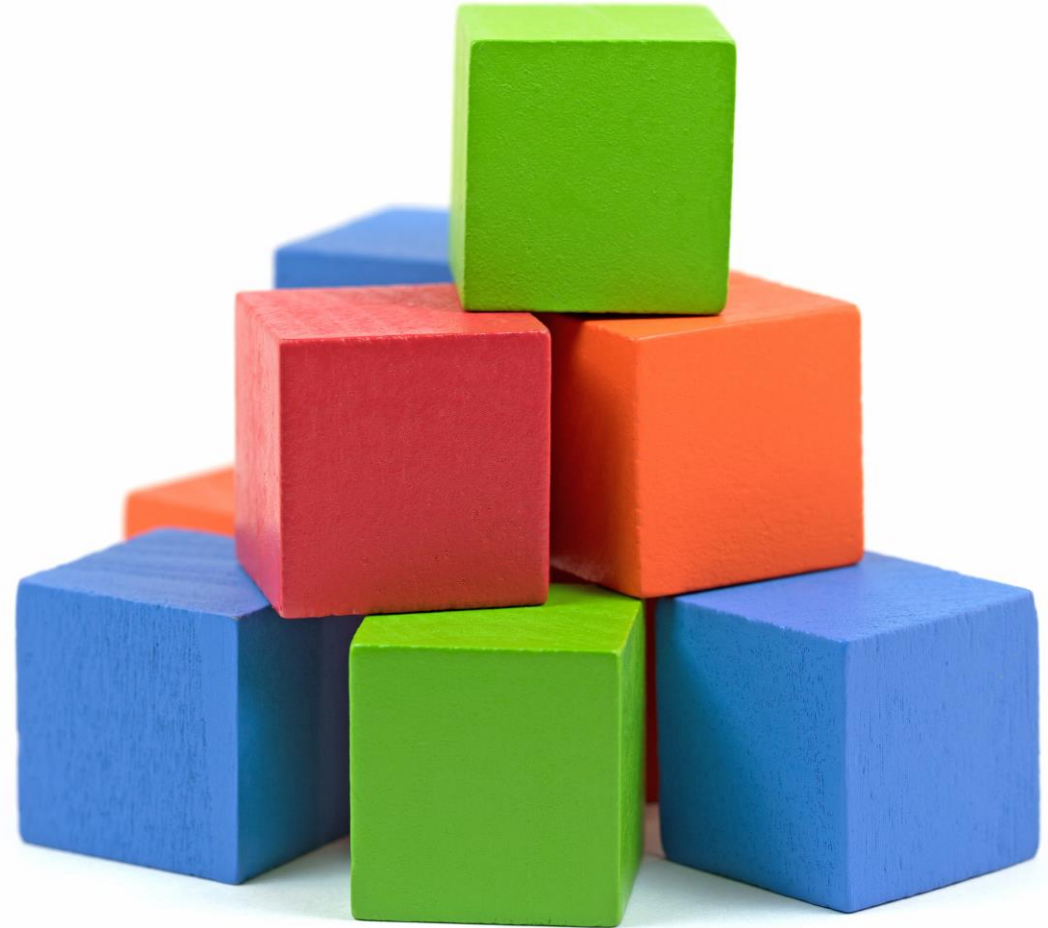
## US National

Adresses

Phone Numbers

Social Security Number

US Licences Plate

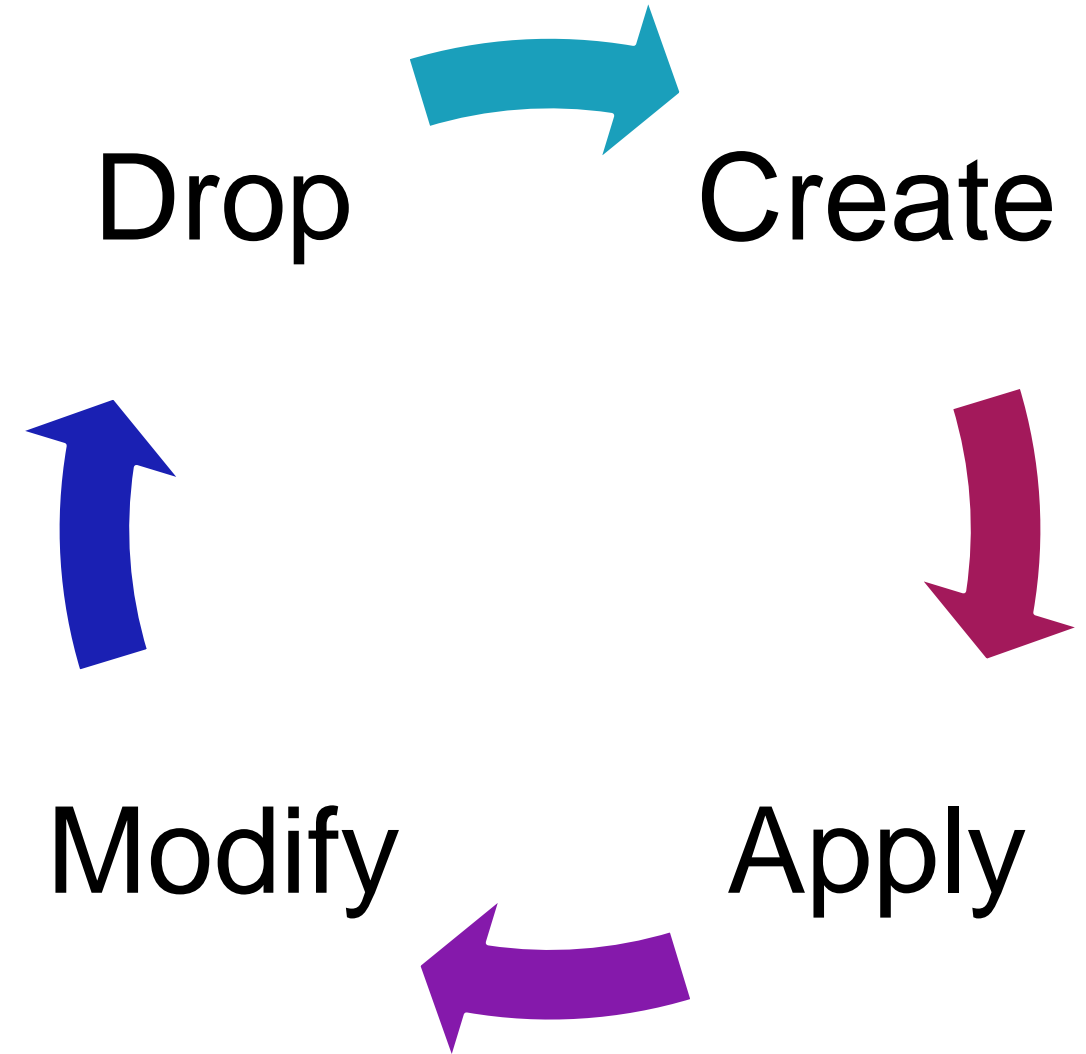




# Domain Lifecycle



# Data Use Case Domain Lifecycle





# Data Use Case Domain: Modfiy

Modify (alter domain)  
options are limited to

Display  
Function

add

modify

drop

Order  
Function

add

modify

drop

Annotations

add

modify

drop





# Data Use Case Domain: Drop & Re-Create

**DROP DOMAIN <name>**

Fails if associated with columns

**drop domain <name> FORCE**

disassociates domain from columns  
removes

- domain check constraints
- domain defaults
- domain annotations

**drop domain <name> force  
PRESERVE**

removes domain  
keeps constraints & defaults on columns  
→ Attention: possible double constraints





# Data Usecase Domains: Demo

# Conclusion Data Intention Language (DIL)





# Data Intention Language - Annotations

## Context

- Key-Value Pairs
- Application Metadata

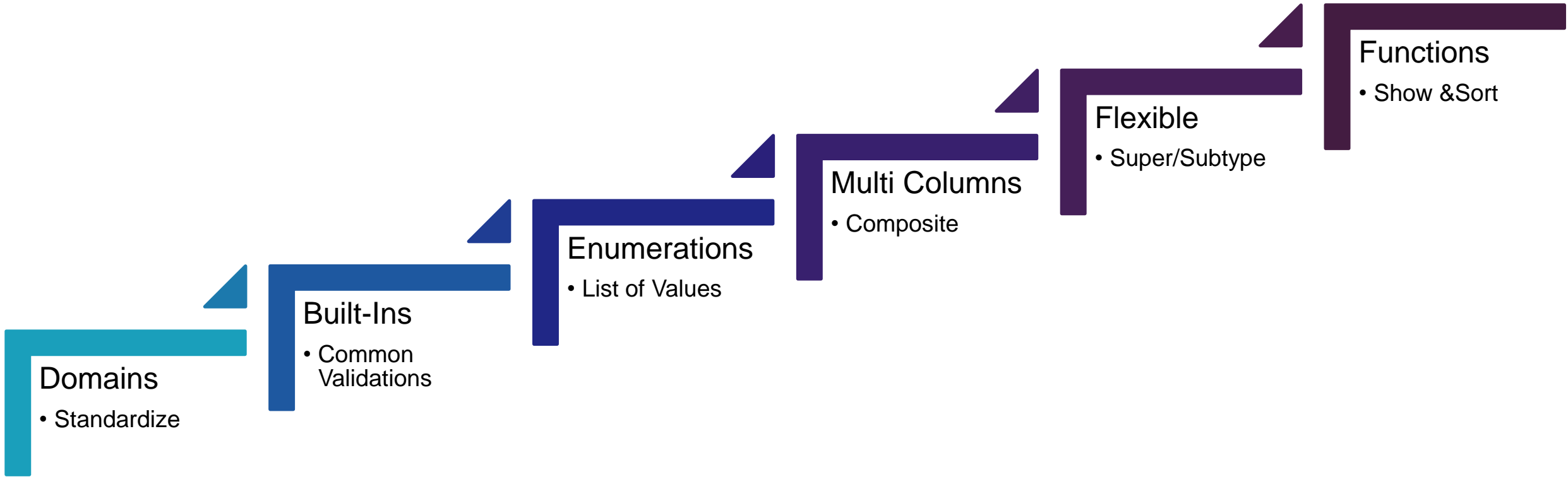
## Understanding your Data

- Devs
- Users
- AI



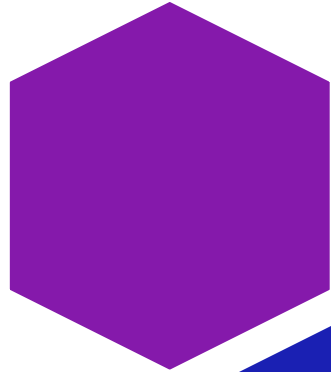


# Data Intention Language - Data Use Case Domains

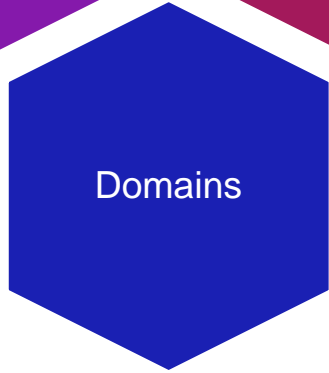




# Data Intention Language: Annotations and Domains



Way more than better comments  
Add Context to your Data



Flexible Tool  
Powerfull with JSON  
Schemas



## ora2know

The German Oracle Database Community

