hroug '23 annual conference

Dynamic attributes to store flexible data

Dr. Andrea Kennel InfoPunkt Kennel GmbH Switzerland

Dr. Andrea Kennel

Consultant

Lecturer for Databases

Coach for Project Management

University of Applied Sciences Northwestern Switzerland

Brugg/Windisch, Switzerland

andrea.kennel@fhnw.ch
andrea@infokennel.ch
www.infokennel.ch

```
{    "type": "router",
     "name": "Hans B4",
     "description": "4 port router",
     "manufacturer": "XYZ",
     "price": 800,
     "port_group": [
       {  "amount": 4,
          "type": "RJ45",
          "speeds": "100/1000"
       },
       {  "amount": 2,
          "type": "SFP",
          "speeds": "1000/10000"
       }
     ],
     "routing": {
       "protocols": "static OSPF",
       "table_size": 5
     }
  }
```

How do the devices look?

```
{    "type": "router",
     "name": "Hans B4",
     "description": "4 port router",
     "manufacturer": "XYZ",
     "price": 800,
     "port_group": [
        {  "amount": 4,
           "type": "RJ45",
           "speeds": "100/1000"
        },
        {  "amount": 2,
           "type": "SFP",
           "speeds": "1000/10000"
        }
     ],
     "routing": {
        "protocols": "static OSPF",
        "table_size": 5
     }
}
```

Could you quickly …

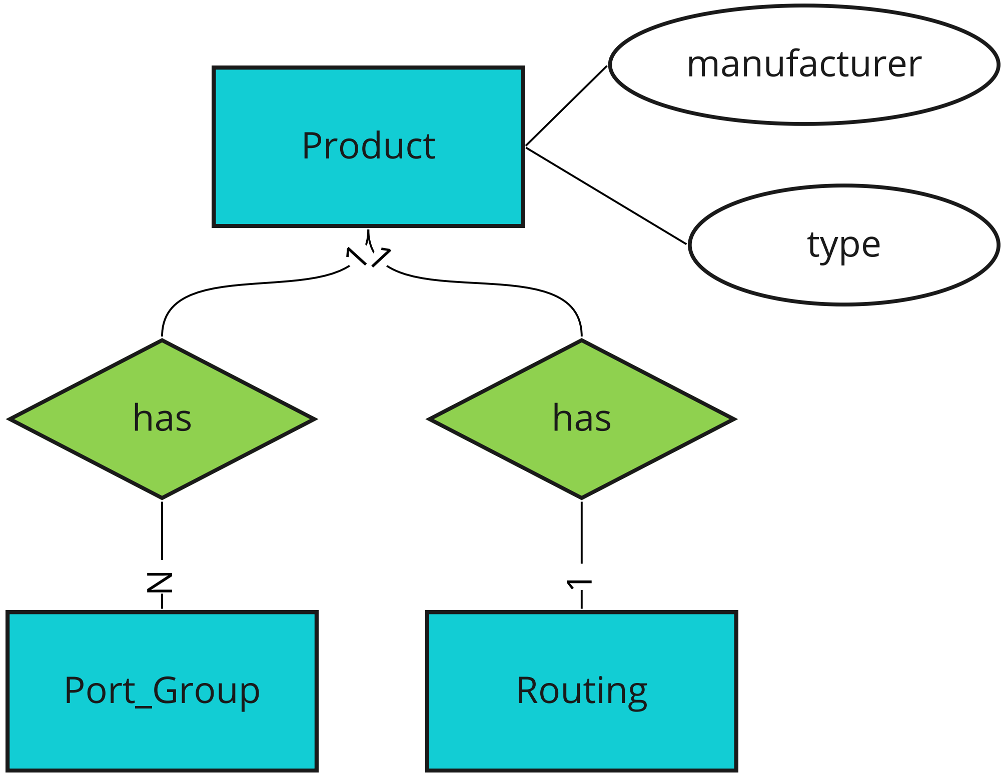
For a web shop that sells network devices, the data on the network devices must be stored in a database.

The data is

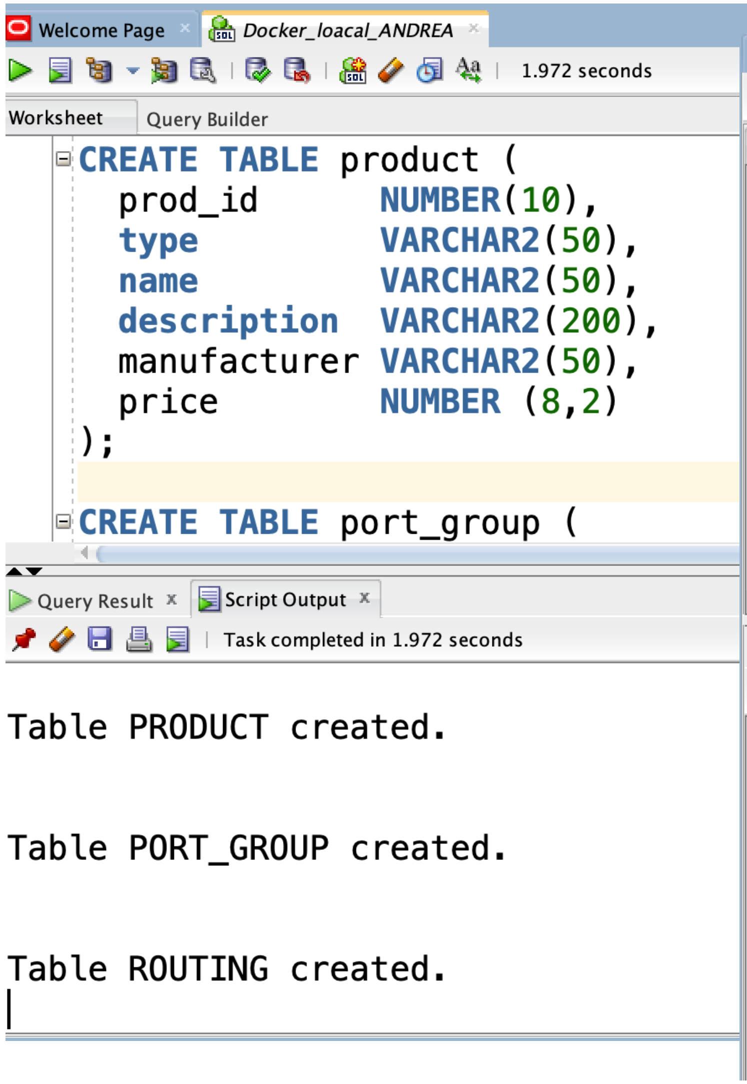
as JSON,

the model is

relatively simple:

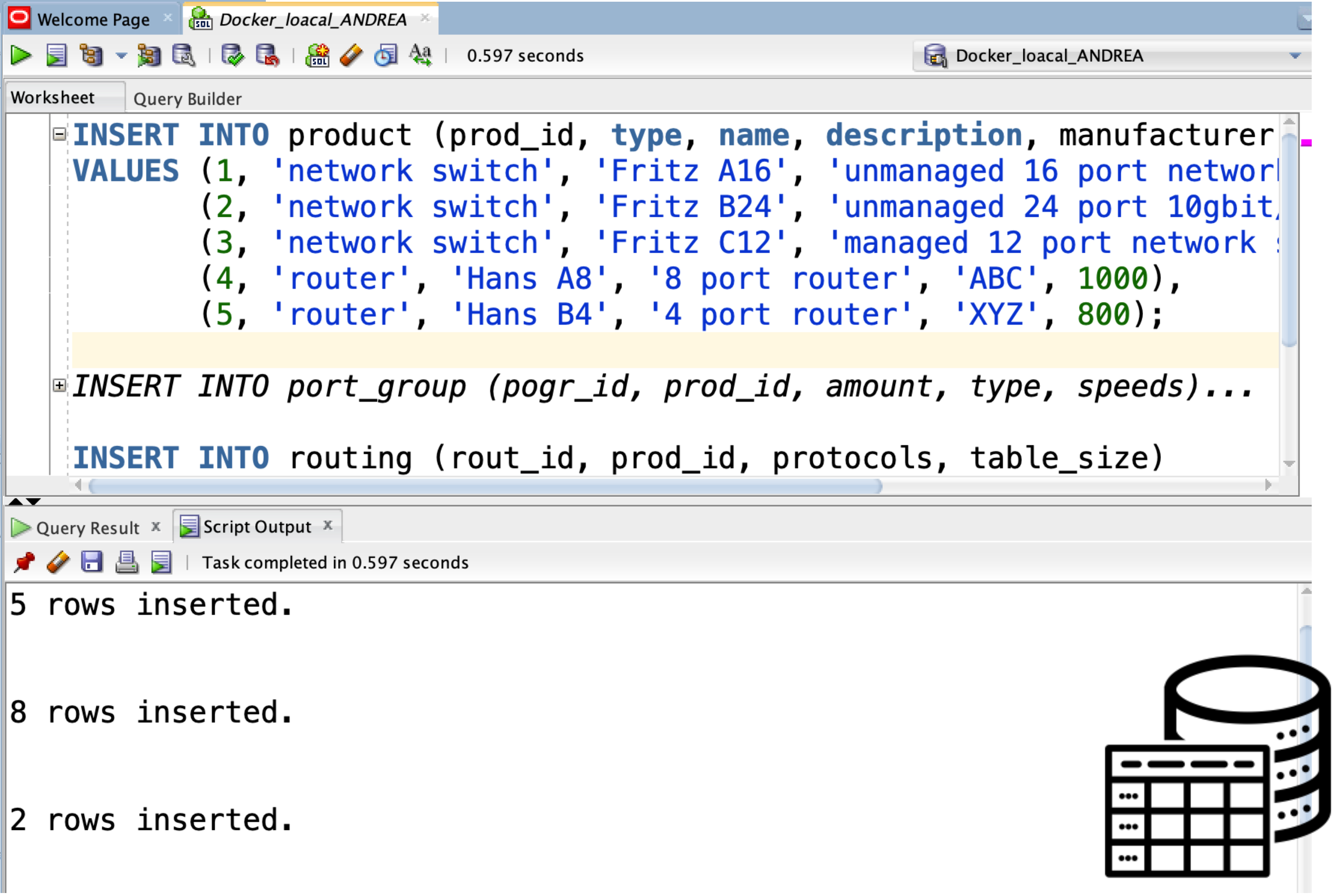


Implement data model and enter data



```
CREATE TABLE product (
  prod_id      NUMBER(10),
  type         VARCHAR2(50),
  name         VARCHAR2(50),
  description  VARCHAR2(200),
  manufacturer VARCHAR2(50),
  price        NUMBER (8,2)
);

CREATE TABLE port_group (
```

```
Table PRODUCT created.


Table PORT_GROUP created.


Table ROUTING created.
```



```
INSERT INTO product (prod_id, type, name, description, manufacturer
VALUES (1, 'network switch', 'Fritz A16', 'unmanaged 16 port networ
       (2, 'network switch', 'Fritz B24', 'unmanaged 24 port 10gbit,
       (3, 'network switch', 'Fritz C12', 'managed 12 port network 
       (4, 'router', 'Hans A8', '8 port router', 'ABC', 1000),
       (5, 'router', 'Hans B4', '4 port router', 'XYZ', 800);

INSERT INTO port_group (pogr_id, prod_id, amount, type, speeds)...

INSERT INTO routing (rout_id, prod_id, protocols, table_size)
```

```
5 rows inserted.


8 rows inserted.


2 rows inserted.
```

We have a new device

```
{   "type": "network switch/layer3 switch",
      "name": "Fritz C16",
      "description": "16 port PoE layer 3 network switch",
      "manufacturer": "ABC",
      "price": 500,
      "port_group": [
         {   "amount": 16,
             "type": "RJ45",
             "speeds": "10/100/1000",
             "poe": {"modes": ["active", "passive"],
                "volt": [24, 48]}
         }
      ],
…
"feature": [
         {   "name": "VLAN",
             "amount": 4094},
         {    "name": "QoS",
             "amount": 8},
         {   "name": "network access control",
             "type": "MAC based authentication",
             "vlan_support": true},
         {   "name": "routing",
             "protocols": "static, RIP, OSPF, BGP",
             "table_size": 10}
     ]
}
```

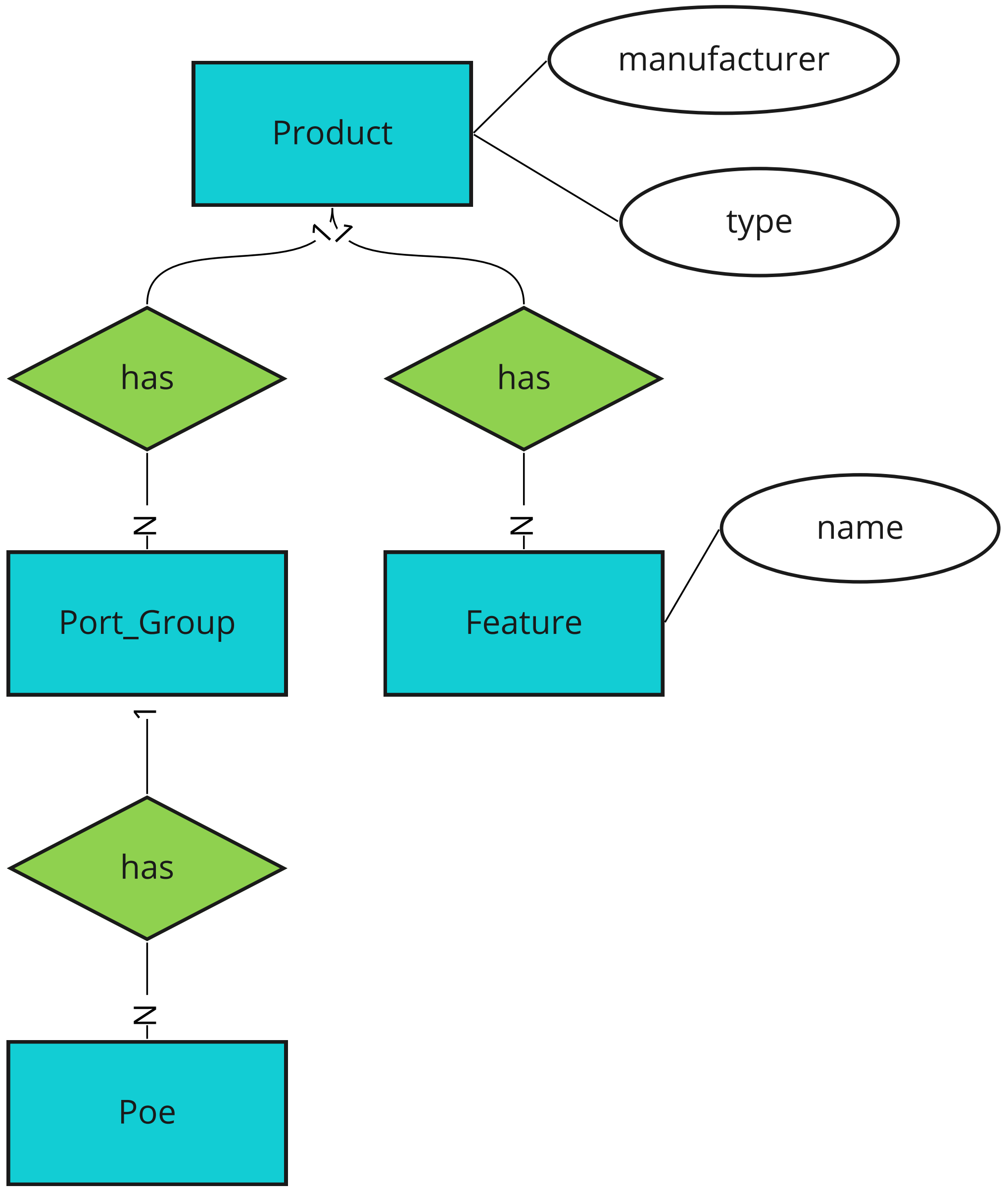
Then we have to adapt the data structure

Detail table for port group and

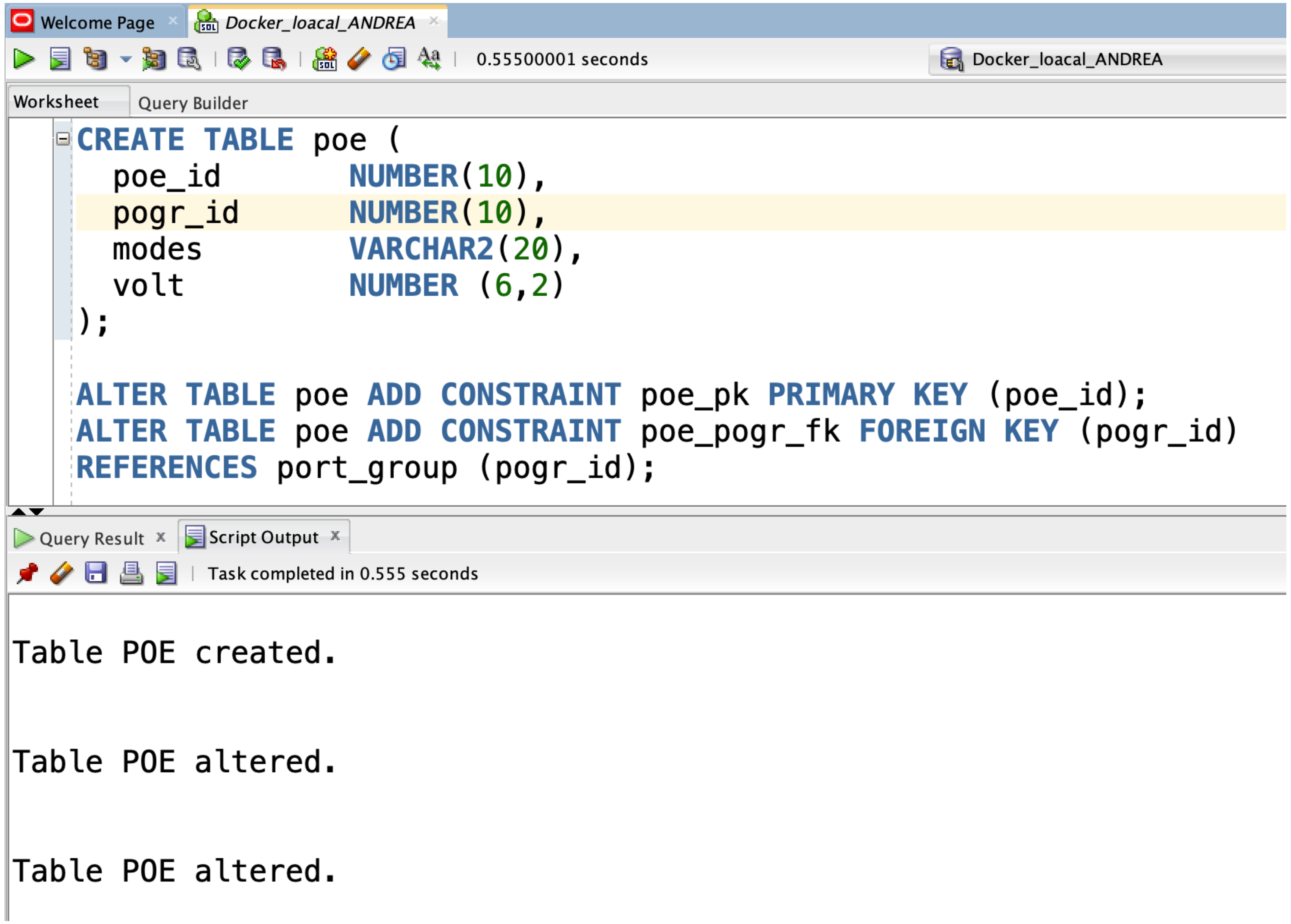
Feature as generalisation with several specialisations

We don't know what else is coming and combine all attributes in the genealisation.

New data structure



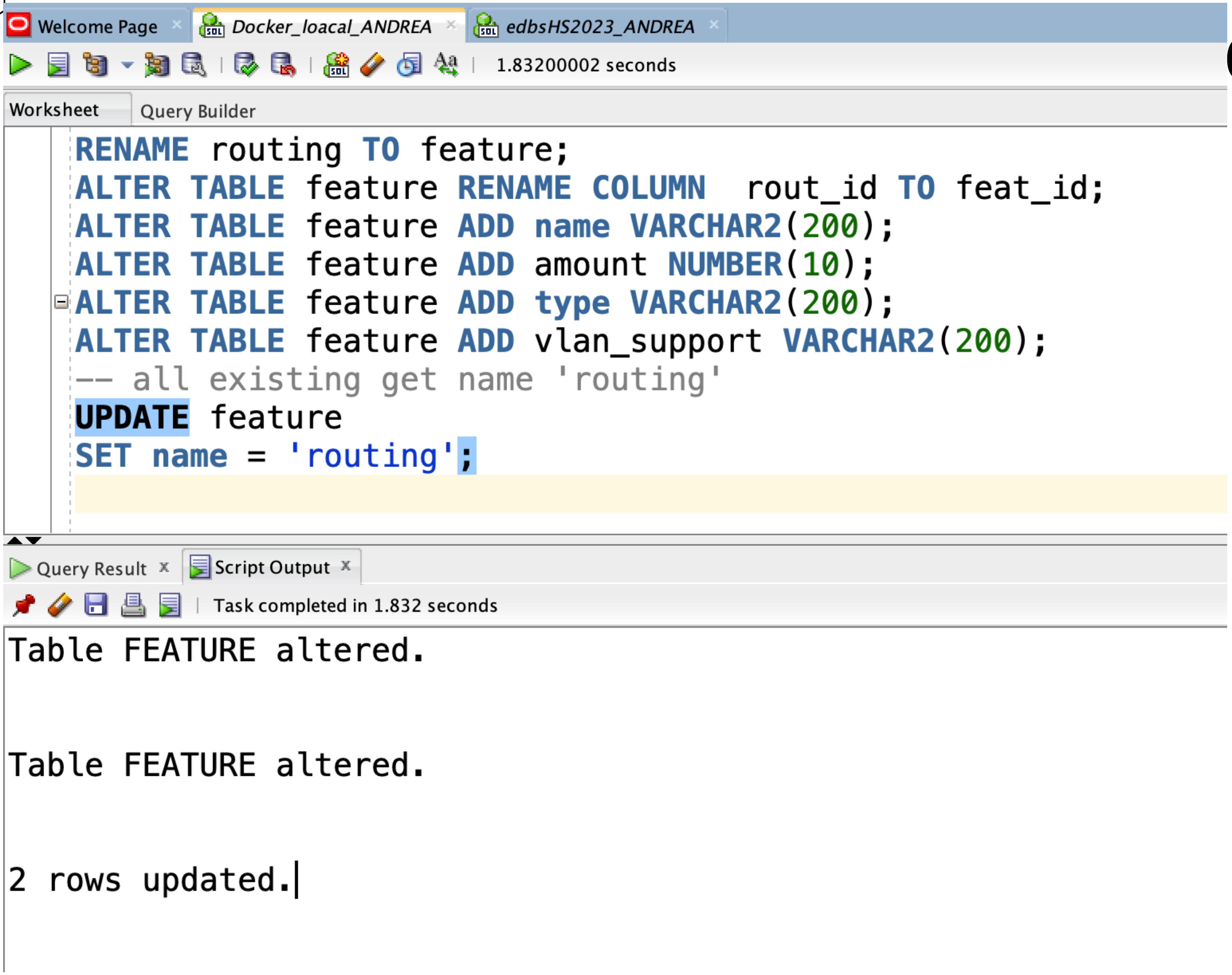
New data structure



```
CREATE TABLE poe (
   poe_id          NUMBER(10),
   pogr_id         NUMBER(10),
   modes           VARCHAR2(20),
   volt            NUMBER (6,2)
);

ALTER TABLE poe ADD CONSTRAINT poe_pk PRIMARY KEY (poe_id);
ALTER TABLE poe ADD CONSTRAINT poe_pogr_fk FOREIGN KEY (pogr_id)
REFERENCES port_group (pogr_id);
```

```
Table POE created.


Table POE altered.


Table POE altered.
```

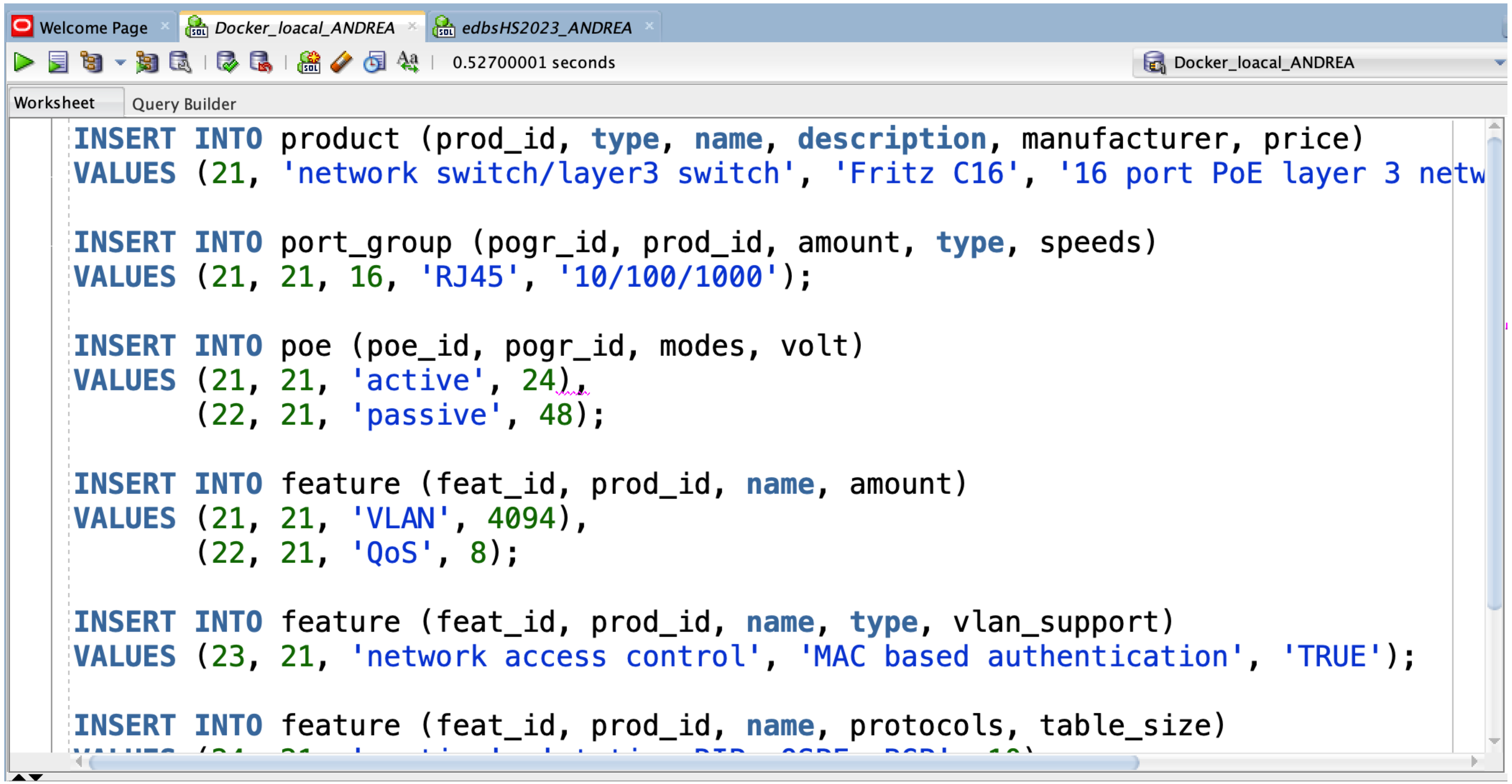
data structure



```sql
RENAME routing TO feature;
ALTER TABLE feature RENAME COLUMN  rout_id TO feat_id;
ALTER TABLE feature ADD name VARCHAR2(200);
ALTER TABLE feature ADD amount NUMBER(10);
ALTER TABLE feature ADD type VARCHAR2(200);
ALTER TABLE feature ADD vlan_support VARCHAR2(200);
-- all existing get name 'routing'
UPDATE feature
SET name = 'routing';
```

```
Table FEATURE altered.


Table FEATURE altered.


2 rows updated.
```

New data



```
INSERT INTO product (prod_id, type, name, description, manufacturer, price)
VALUES (21, 'network switch/layer3 switch', 'Fritz C16', '16 port PoE layer 3 netw

INSERT INTO port_group (pogr_id, prod_id, amount, type, speeds)
VALUES (21, 21, 16, 'RJ45', '10/100/1000');

INSERT INTO poe (poe_id, pogr_id, modes, volt)
VALUES (21, 21, 'active', 24),
       (22, 21, 'passive', 48);

INSERT INTO feature (feat_id, prod_id, name, amount)
VALUES (21, 21, 'VLAN', 4094),
       (22, 21, 'QoS', 8);

INSERT INTO feature (feat_id, prod_id, name, type, vlan_support)
VALUES (23, 21, 'network access control', 'MAC based authentication', 'TRUE');

INSERT INTO feature (feat_id, prod_id, name, protocols, table_size)
```

What if there are more attributes?

We had to add more attributes to FEATURES.

What happens if there are more attributes?

What if many features only have a few attributes filled?

Is there a more flexible solution?

```
"feature": [
          {   "name": "VLAN",
              "amount": 4094},
          {    "name": "QoS",
              "amount": 8},
          {   "name": "network access control",
              "type": "MAC based authentication",
              "vlan_support": true},
          {   "name": "routing",
              "protocols": "static, RIP, OSPF, BGP",
              "table_size": 10}
    ]
}
```

EAV

What if there are more attributes?

The answer is EAV

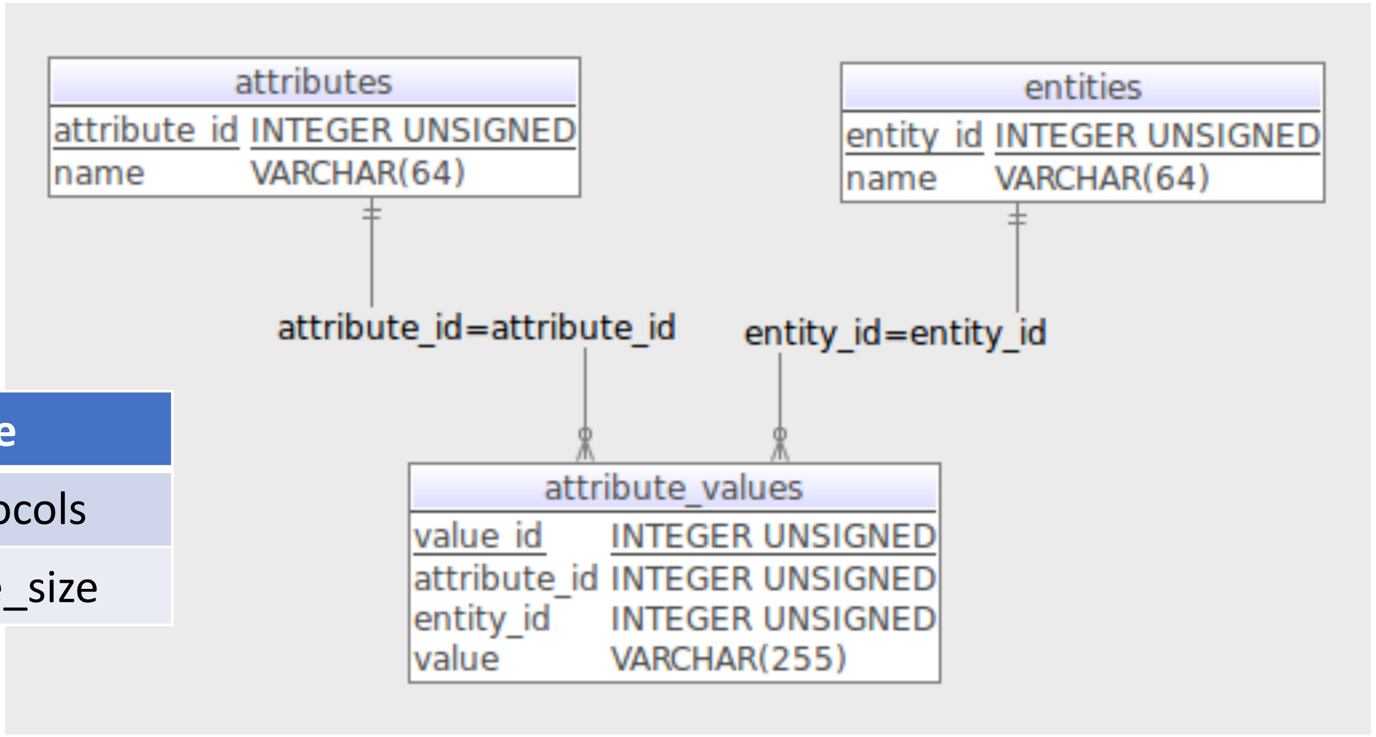
The Entity Attribute Value (EAV) model is a data modelling technique used in databases to store and retrieve data in a flexible and scalable way.

Details about the model: https://inviqa.com/blog/understanding-eav-data-model-and-when-use-it

EAV

EAV

Attribute_id	name
84	protocols
85	table_size

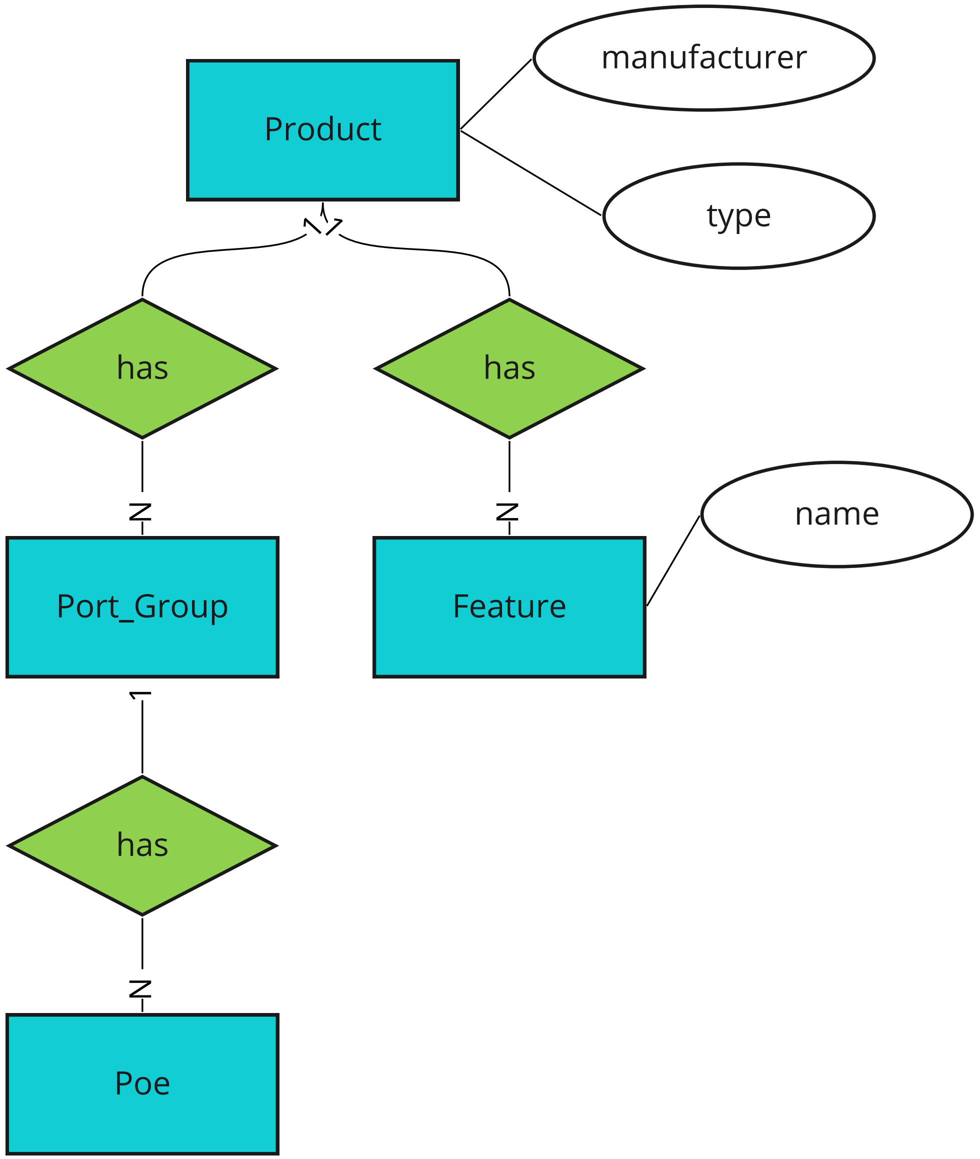

attributes
attribute_id INTEGER UNSIGNED
name VARCHAR(64)
entities
entity_id INTEGER UNSIGNED
name VARCHAR(64)
attribute_id=attribute_id
entity_id=entity_id
attribute_values
value_id INTEGER UNSIGNED
attribute_id INTEGER UNSIGNED
entity_id INTEGER UNSIGNED
value VARCHAR(255)

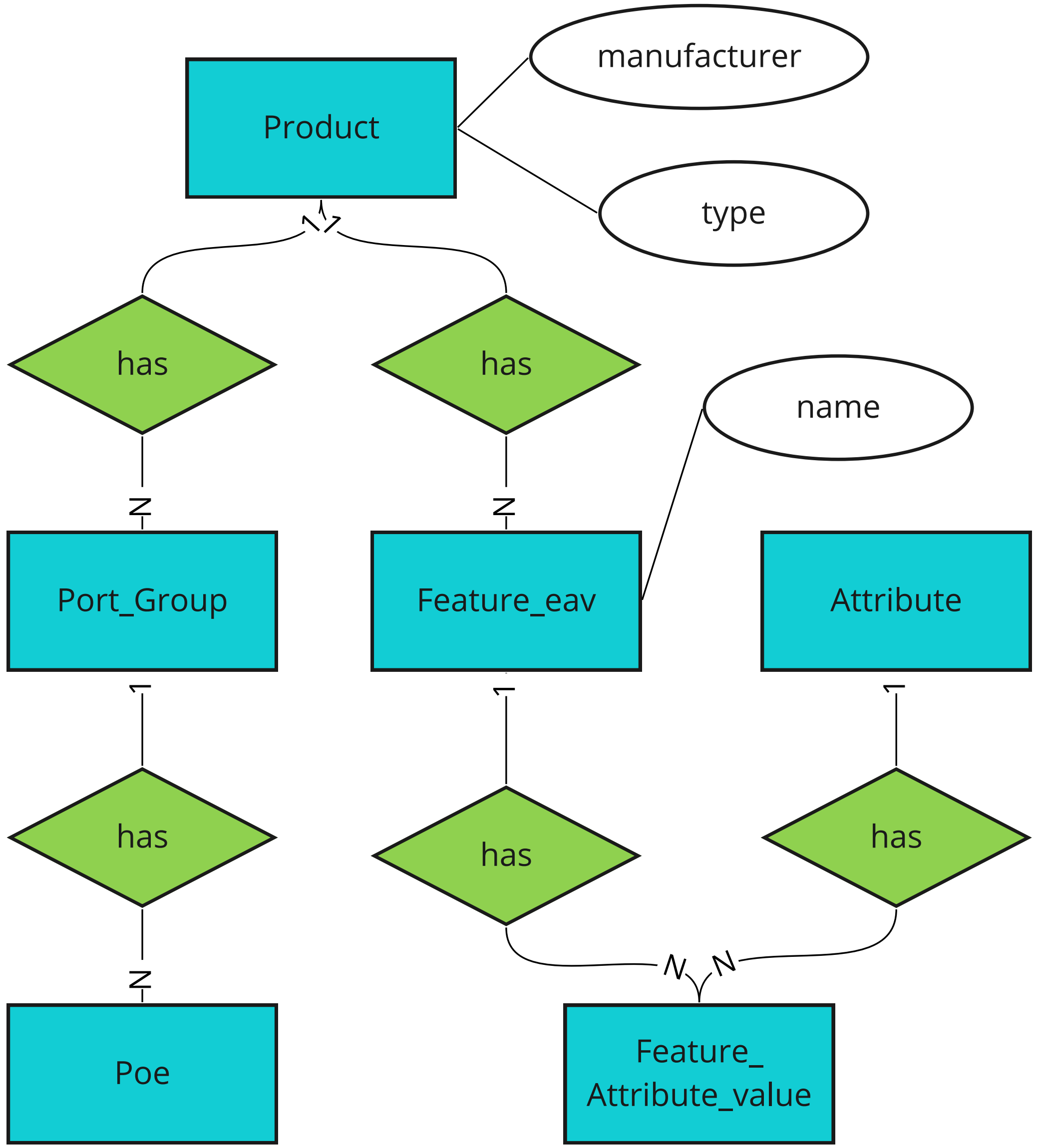
```
INSERT INTO feature (feat_id, prod_id, name, protocols, table_size)
VALUES (24, 21, 'routing', 'static, RIP, OSPF, BGP', 10);
```

Value_id	Attribute_id	Entity_id	values
1	84	24	static, RIP, OSPF, BGP
2	85	24	10

EAV

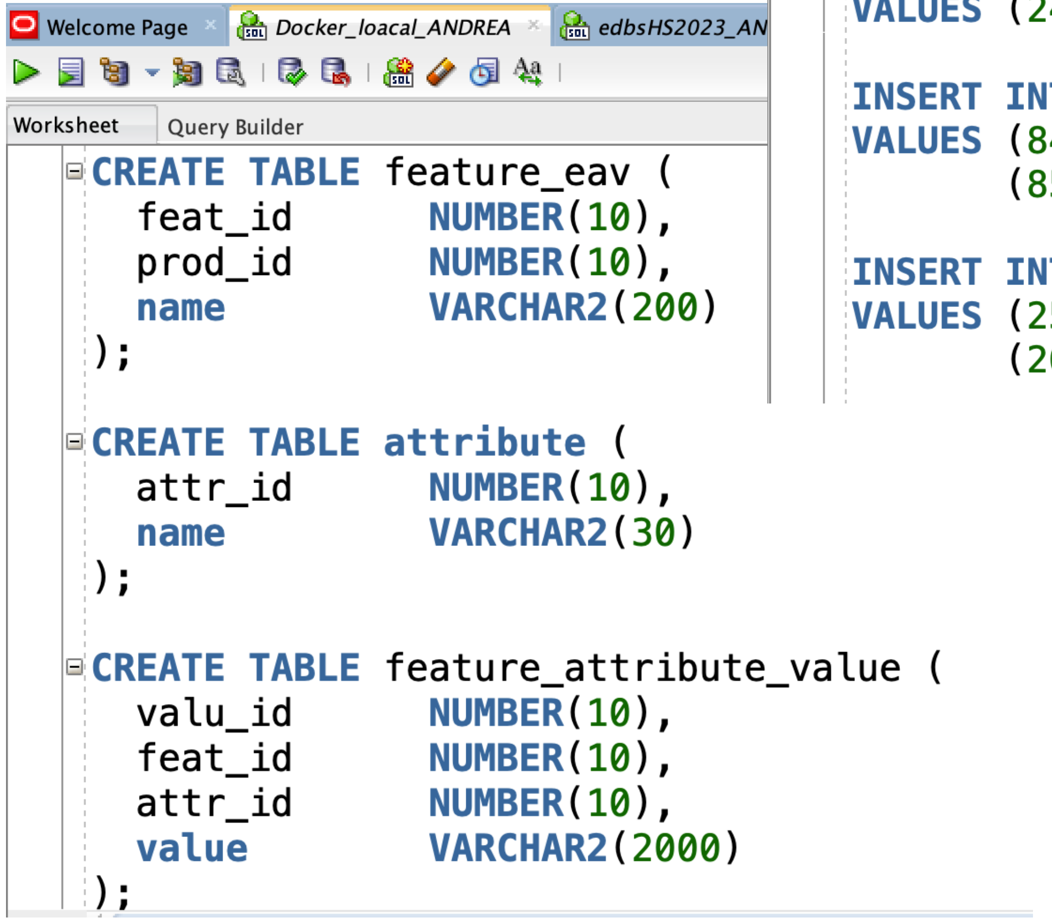
EAV


Product
manufacturer
type
has
has
N
N
Port_Group
Feature
name
1
has
N
Poe

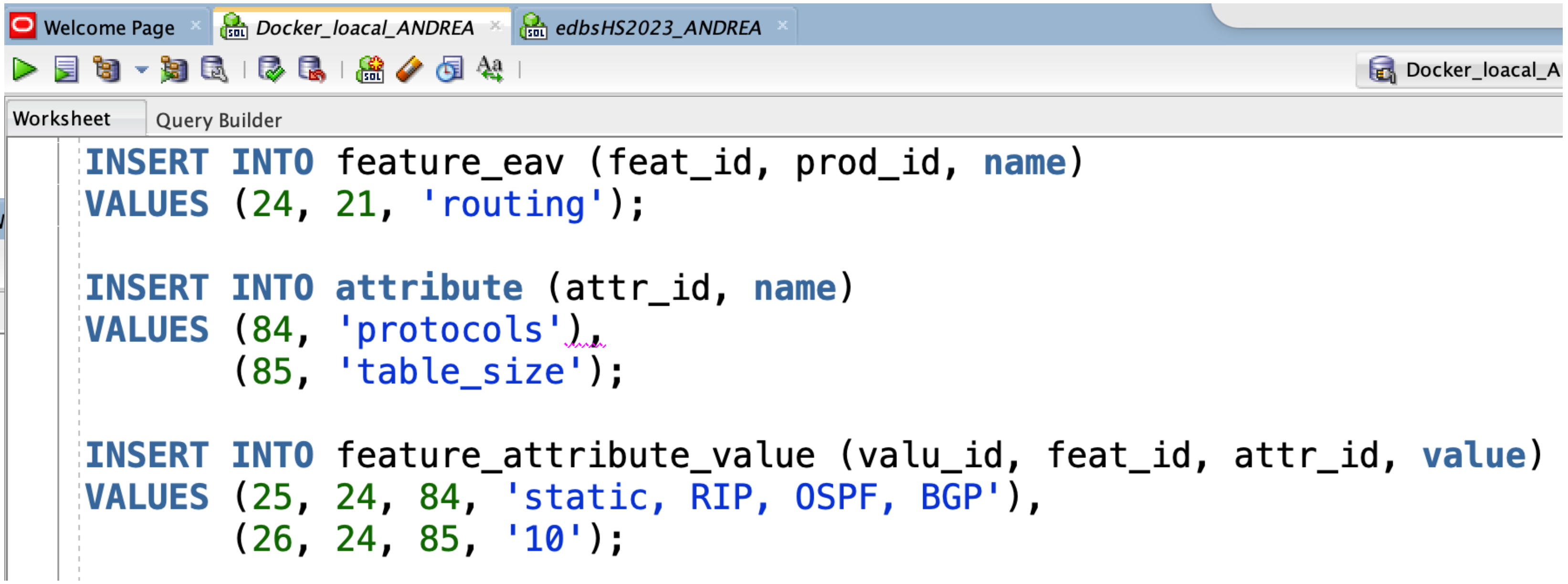

Product
manufacturer
type
has
has
N
N
Port_Group
Feature_eav
name
Attribute
1
1
1
has
has
has
N
N
N
Poe
Feature_
Attribute_value

EAV

EAV



```
CREATE TABLE feature_eav (
  feat_id       NUMBER(10),
  prod_id       NUMBER(10),
  name          VARCHAR2(200)
);

CREATE TABLE attribute (
  attr_id       NUMBER(10),
  name          VARCHAR2(30)
);

CREATE TABLE feature_attribute_value (
  valu_id       NUMBER(10),
  feat_id       NUMBER(10),
  attr_id       NUMBER(10),
  value         VARCHAR2(2000)
);
```



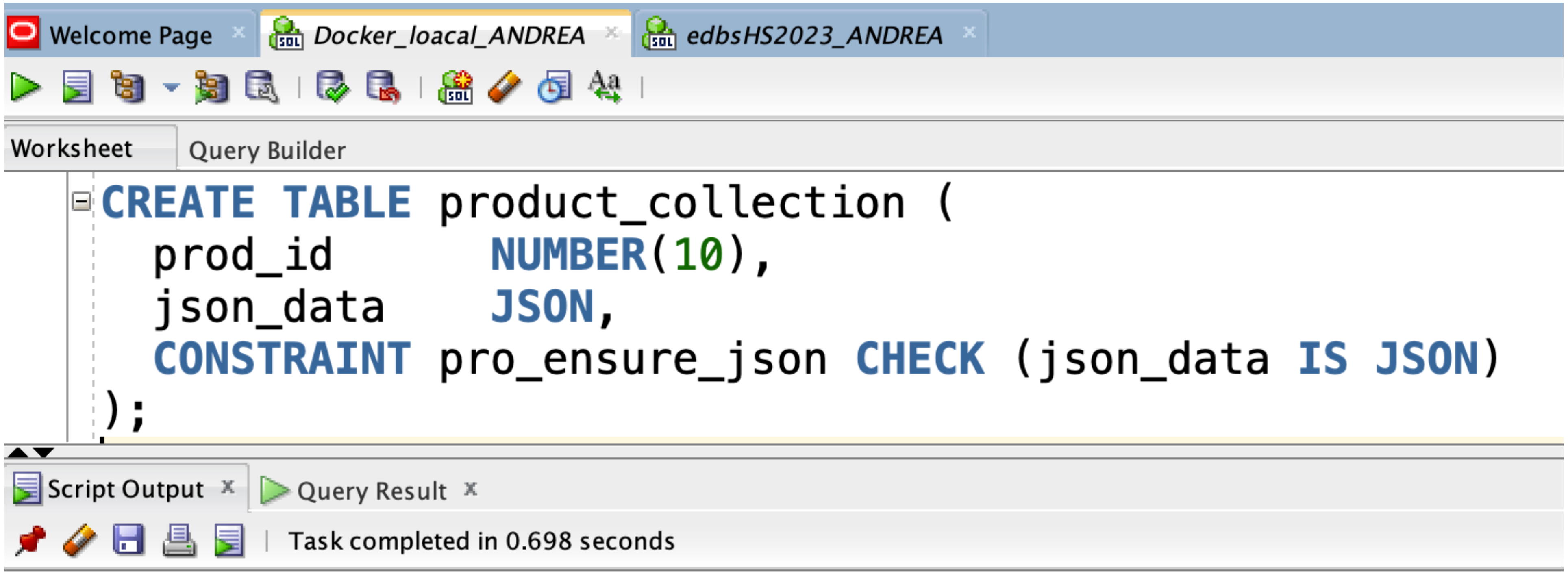
```
INSERT INTO feature_eav (feat_id, prod_id, name)
VALUES (24, 21, 'routing');

INSERT INTO attribute (attr_id, name)
VALUES (84, 'protocols'),
       (85, 'table_size');

INSERT INTO feature_attribute_value (valu_id, feat_id, attr_id, value)
VALUES (25, 24, 84, 'static, RIP, OSPF, BGP'),
       (26, 24, 85, '10');
```

EAV

Wouldn't a document DB be better?

- I am getting confused
- Why not just store all the data as JSON?

Wouldn't a document DB be better?

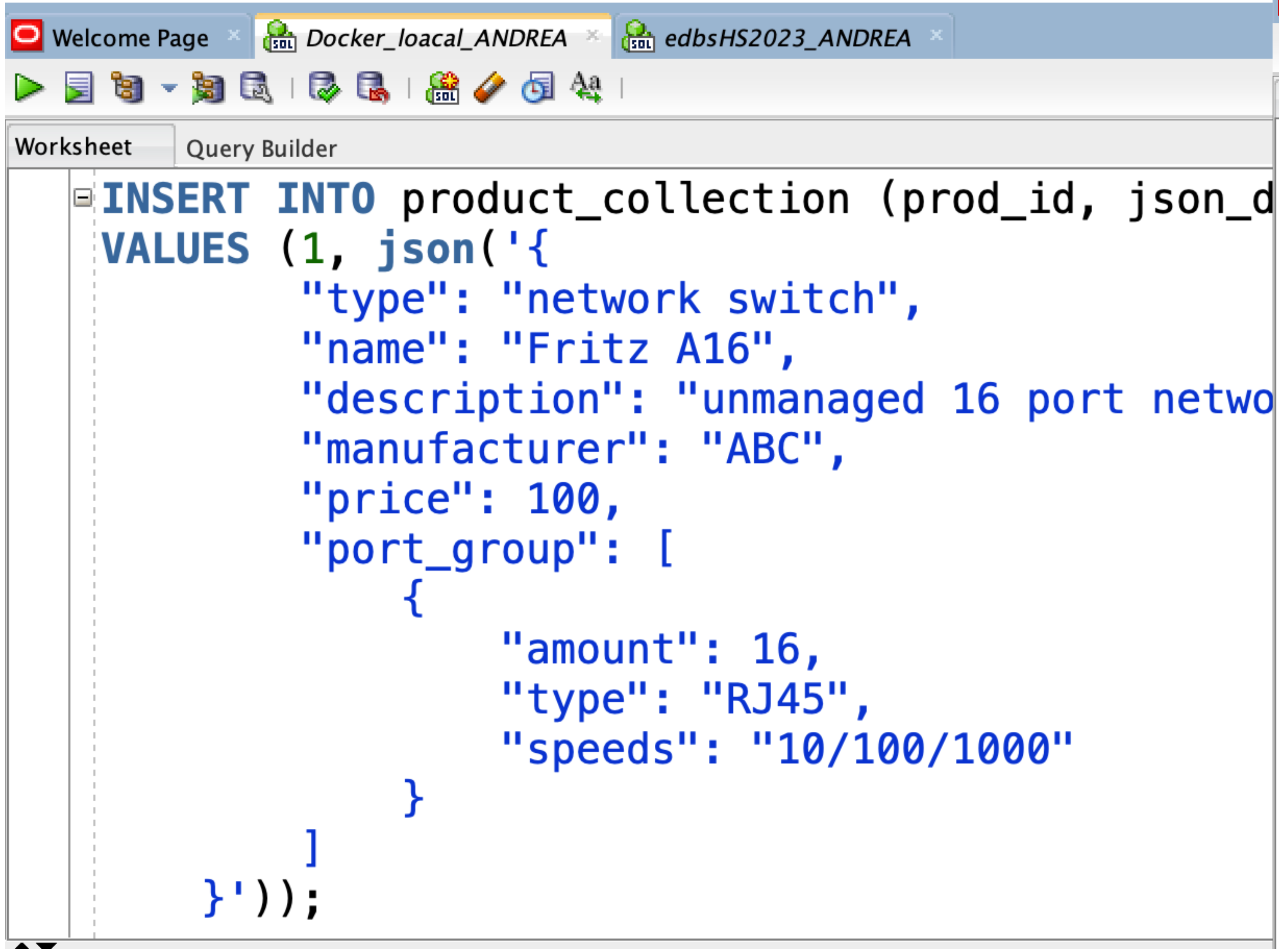


```
CREATE TABLE product_collection (
  prod_id       NUMBER(10),
  json_data     JSON,
  CONSTRAINT pro_ensure_json CHECK (json_data IS JSON)
);
```

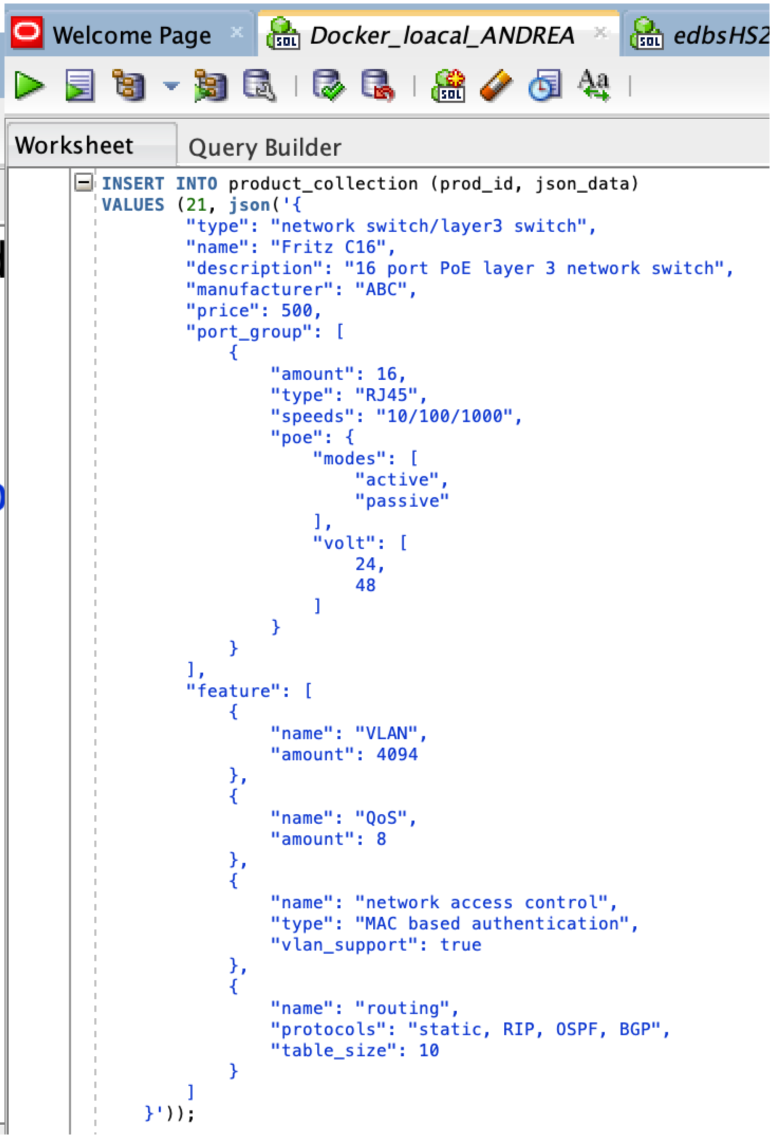
Task completed in 0.698 seconds

```
Table PRODUCT_COLLECTION created.
```

Wouldn't a document DB be better?



```
INSERT INTO product_collection (prod_id, json_d
VALUES (1, json('{
            "type": "network switch",
            "name": "Fritz A16",
            "description": "unmanaged 16 port netwo
            "manufacturer": "ABC",
            "price": 100,
            "port_group": [
                {
                    "amount": 16,
                    "type": "RJ45",
                    "speeds": "10/100/1000"
                }
            ]
      }'));
```

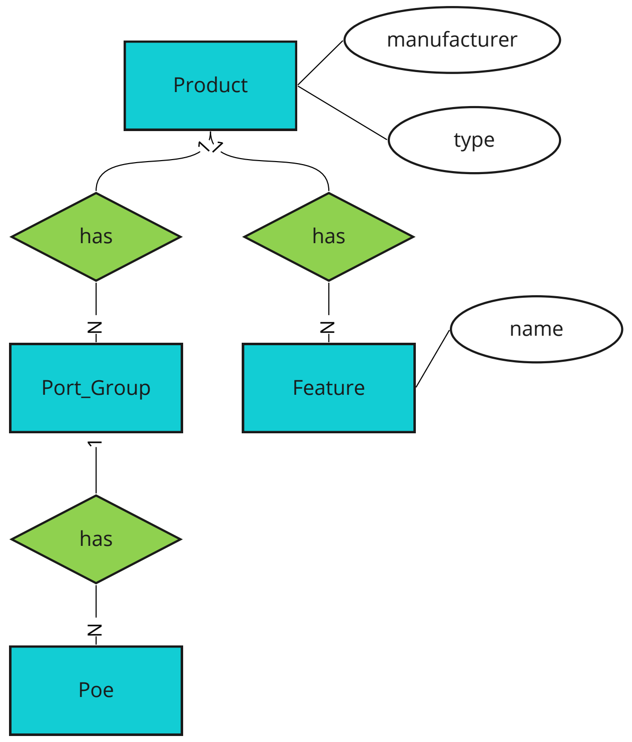


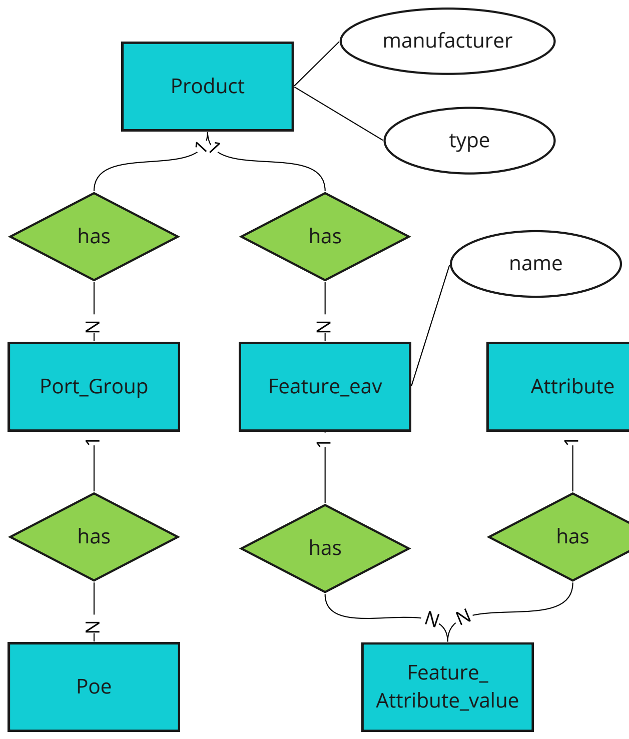
```
INSERT INTO product_collection (prod_id, json_data)
VALUES (21, json('{
        "type": "network switch/layer3 switch",
        "name": "Fritz C16",
        "description": "16 port PoE layer 3 network switch",
        "manufacturer": "ABC",
        "price": 500,
        "port_group": [
            {
                "amount": 16,
                "type": "RJ45",
                "speeds": "10/100/1000",
                "poe": {
                    "modes": [
                        "active",
                        "passive"
                    ],
                    "volt": [
                        24,
                        48
                    ]
                }
            }
        ],
        "feature": [
            {
                "name": "VLAN",
                "amount": 4094
            },
            {
                "name": "QoS",
                "amount": 8
            },
            {
                "name": "network access control",
                "type": "MAC based authentication",
                "vlan_support": true
            },
            {
                "name": "routing",
                "protocols": "static, RIP, OSPF, BGP",
                "table_size": 10
            }
        ]
    }'));
```

Wouldn't a document DB be better?

What does it look like with queries?

Example: Do we have devices with a port group of type SFP?





Do we have devices with a port group of type SFP? relational

```
SELECT pro.prod_id, pro.type product_type, pro.name, pog.amount, pog.type
FROM product pro INNER JOIN
    port_group pog ON (pro.prod_id = pog.prod_id)
WHERE pog.type = 'SFP’;
```

Do we have devices with a port group of type SFP? JSON

```
SELECT pc.prod_id, pc.json_data.type product_type,
pc.json_data.name,
      pc.json_data.port_group[*].amount,
      pc.json_data.port_group[*].type
FROM product_collection pc
WHERE json_exists(pc.json_data,
                '$.port_group?(@.type == $v1)'
                 PASSING 'SFP' AS "v1");
```

```
SELECT pc.prod_id, pc.json_data.type product_type,
pc.json_data.name,
      jt.*
FROM product_collection pc,
    JSON_TABLE(
        pc.json_data
        COLUMNS (
            NESTED port_group[*]
            COLUMNS (
                amount NUMBER(3) PATH '$.amount',
                type VARCHAR2(50) PATH '$.type'
            )
        )
    ) jt
WHERE jt.type = 'SFP';
```

Do we have feature devices that have amount = 8? relational

```
SELECT pro.prod_id, pro.type product_type, pro.name,
       fe.name feature_name, fe.amount
FROM product pro INNER JOIN
     feature fe ON (pro.prod_id = fe.prod_id)
WHERE fe.amount = 8;
```

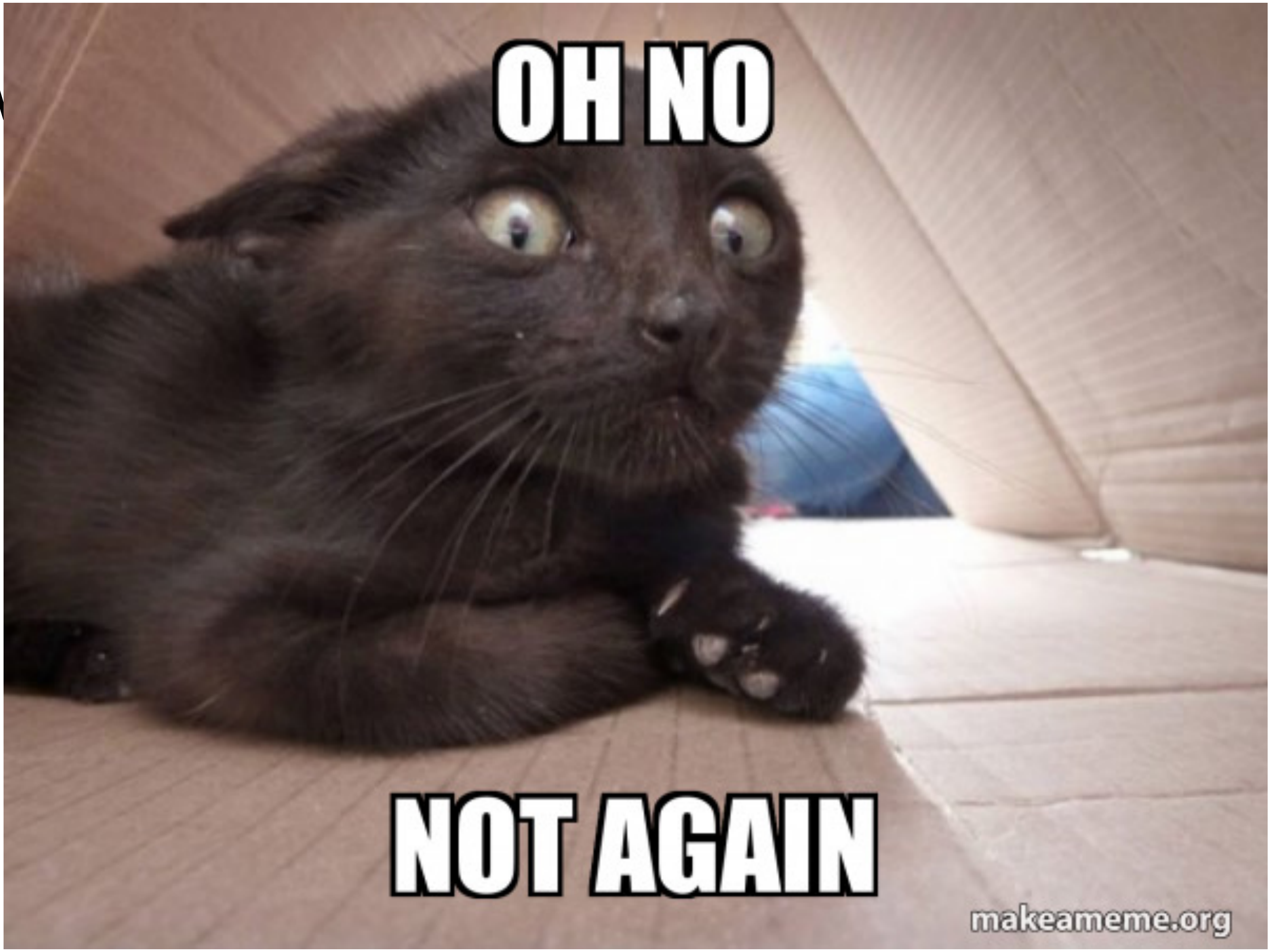
Do we have feature devices that have amount = 8? JSON

```
SELECT pc.prod_id, pc.json_data.type product_type, pc.json_data.name,
     jt.*
FROM product_collection pc,
    JSON_TABLE(
       pc.json_data
       COLUMNS (
          NESTED feature[*]
          COLUMNS (
             feature_name VARCHAR2(50) PATH '$.name',
             amount NUMBER(10) PATH '$.amount'
          )
       )
    ) jt
WHERE  jt.amount = 8;
```

Do we have feature devices that have amount = 8? EAV

```
SELECT pro.prod_id, pro.type product_type, pro.name,
      fe.name feature_name, feat.value amount
FROM product pro INNER JOIN
    feature_eav fe ON (pro.prod_id = fe.prod_id) INNER JOIN
    feature_attribute_value feat ON (fe.feat_id = feat.feat_id) INNER JOIN
    attribute at ON (feat.attr_id = at.attr_id)
WHERE at.name = 'amount'
  AND feat.value = '8';
```

We have a new



https://makeameme.org/meme/oh-no-not-6bc4c7

We have a new device again

```
 {   "type": "router/firewall",
      "name": "Hans A48",
....
 "feature": [
          {   … },
          {   … },
          {   … },
          {   "name": "firewall",
              "type": "stateful, stateless",
              "amount_of_rules": 1000,
              "amount_of_connections": 100000,
              "amount_of_nat_rules": 1000,
              "feature": [
                 {   "name": "URL filtering",
                     "types": "whitelist, blacklist",
                     "amount_of_rules": 1000 },
....
```

```
                 {   "name": "application filtering",
                     "types": "whitelist",
                     "amount_of_rules": 1000 },
                 {   "name": "content filtering",
                     "types": "blacklist",
                     "amount_of_rules": 1000 },
                 {   "name": "anti-virus",
                     "ssl_inspection": true }
              ]
          },
{   "name": "VPN",
              "type": "IPsec, SSL, L2TP, PPTP",
              "amount_of_tunnels": 1000,
              "amount_of_users": 1000,
              "feature": [
                 {   "name": "IPsec",
                     "type": "IKEv1, IKEv2",
                     "amount_of_tunnels": 1000,
                     "amount_of_users": 1000 },
                 {   "name": "SSL",
…
```

Then we have to adapt the data structure

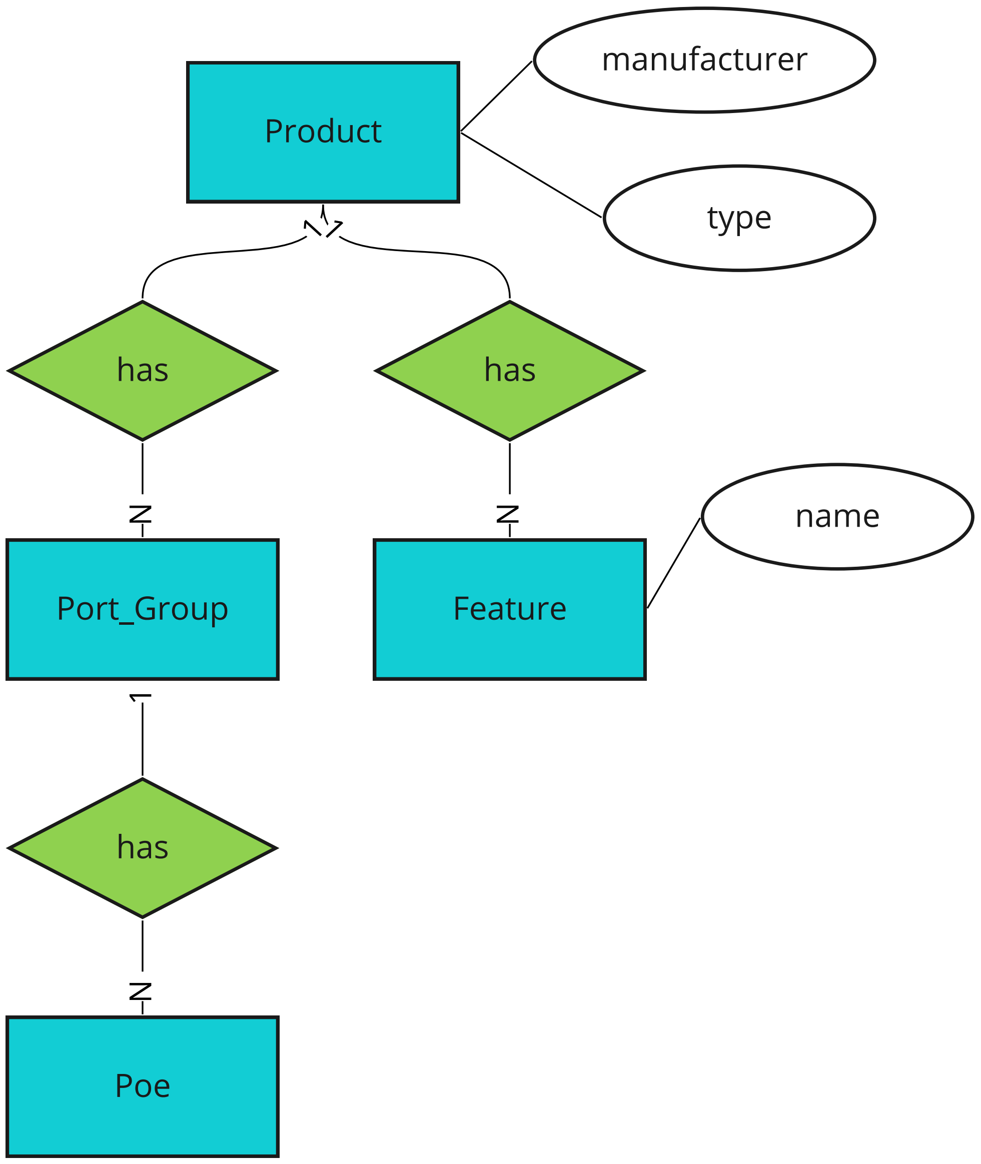
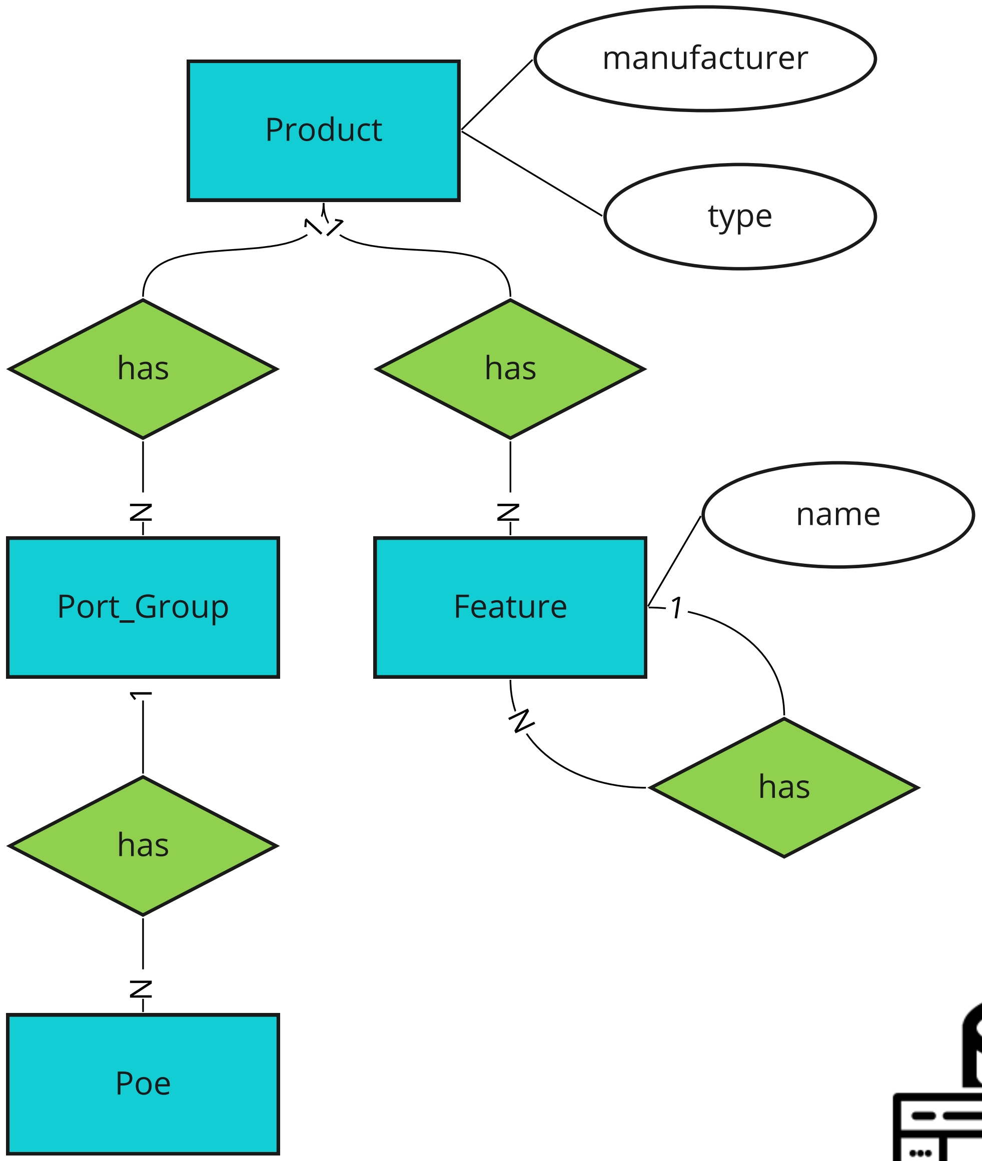
Further types of features with new attributes, we have already expected that.

Features can contain features, that's new, but actually also just another attribute in features.

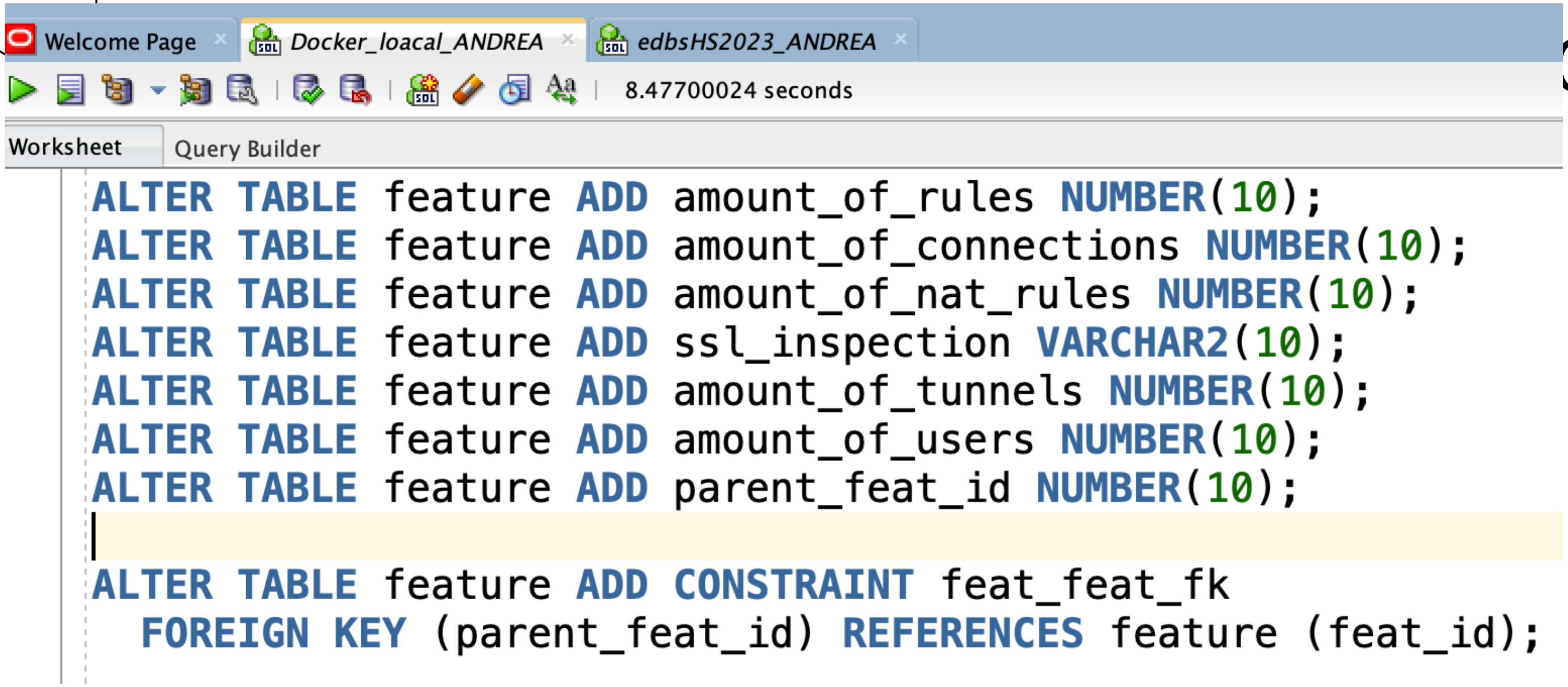
Data model for specific tables

Data model for eav approach

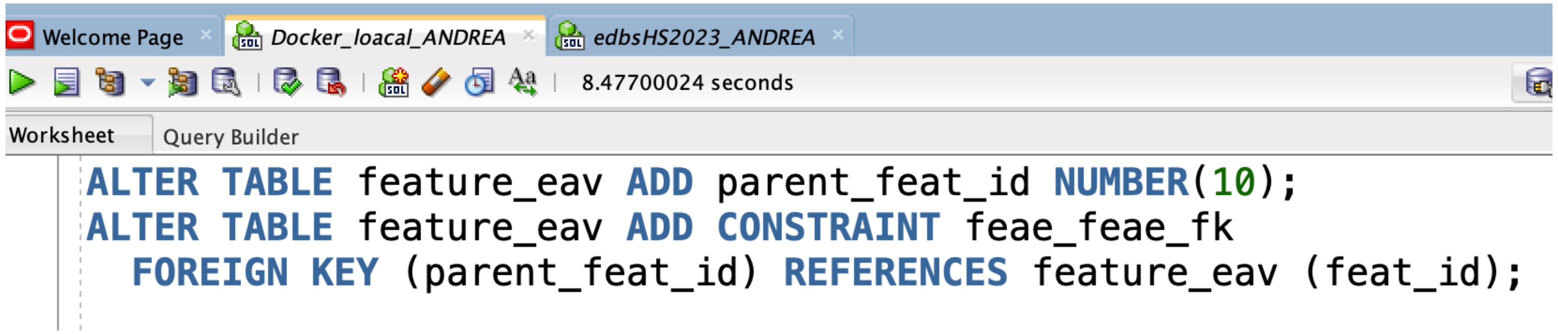
EAV


Product
manufacturer
type
N
has
has
N
N
Port_Group
Feature
name
1
has
N
Poe

Product
manufacturer
type
N
has
has
N
N
Port_Group
Feature
name
1
1
N
has
has
N
Poe

data structure



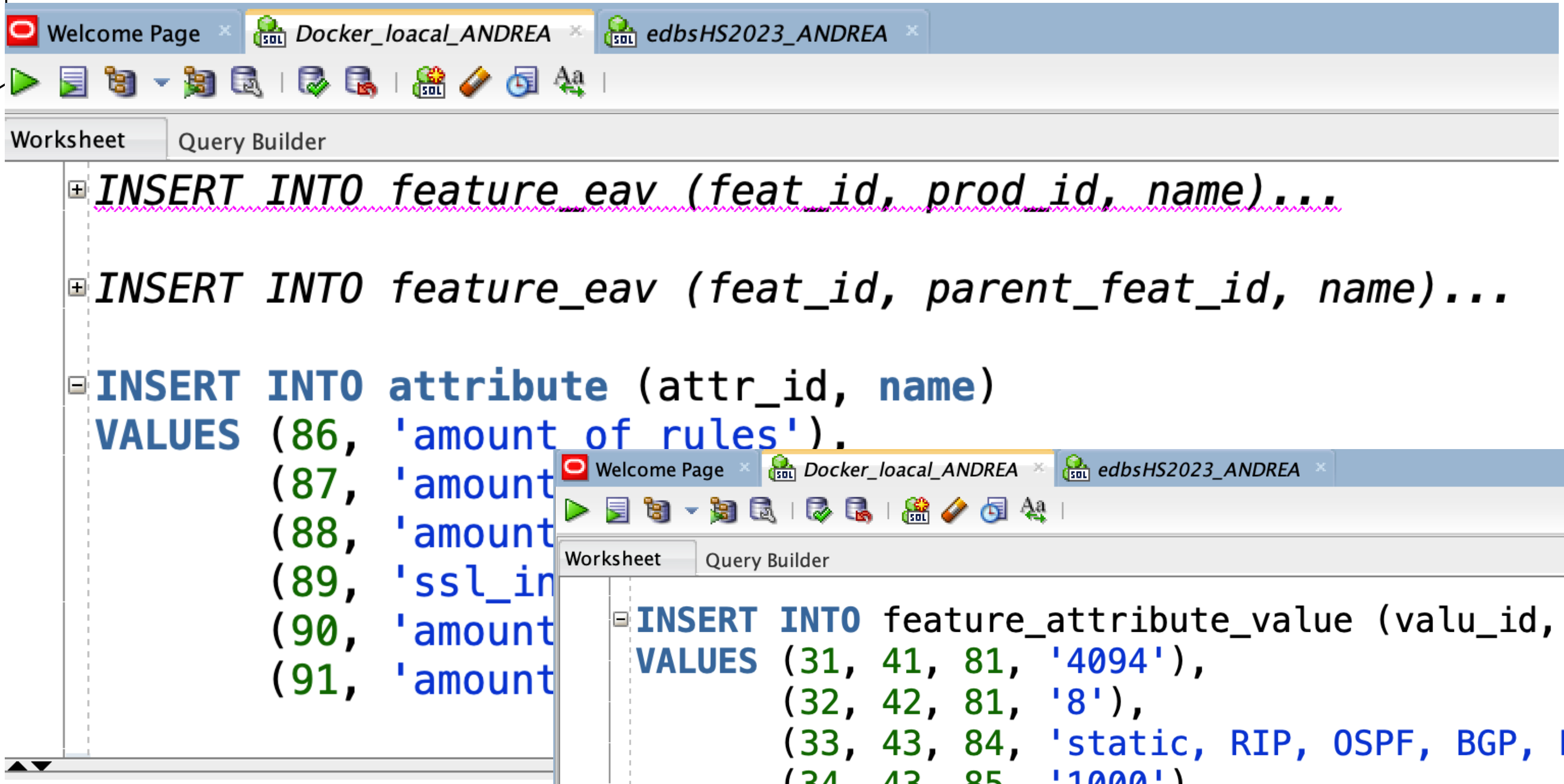
```
ALTER TABLE feature ADD amount_of_rules NUMBER(10);
ALTER TABLE feature ADD amount_of_connections NUMBER(10);
ALTER TABLE feature ADD amount_of_nat_rules NUMBER(10);
ALTER TABLE feature ADD ssl_inspection VARCHAR2(10);
ALTER TABLE feature ADD amount_of_tunnels NUMBER(10);
ALTER TABLE feature ADD amount_of_users NUMBER(10);
ALTER TABLE feature ADD parent_feat_id NUMBER(10);

ALTER TABLE feature ADD CONSTRAINT feat_feat_fk
  FOREIGN KEY (parent_feat_id) REFERENCES feature (feat_id);
```



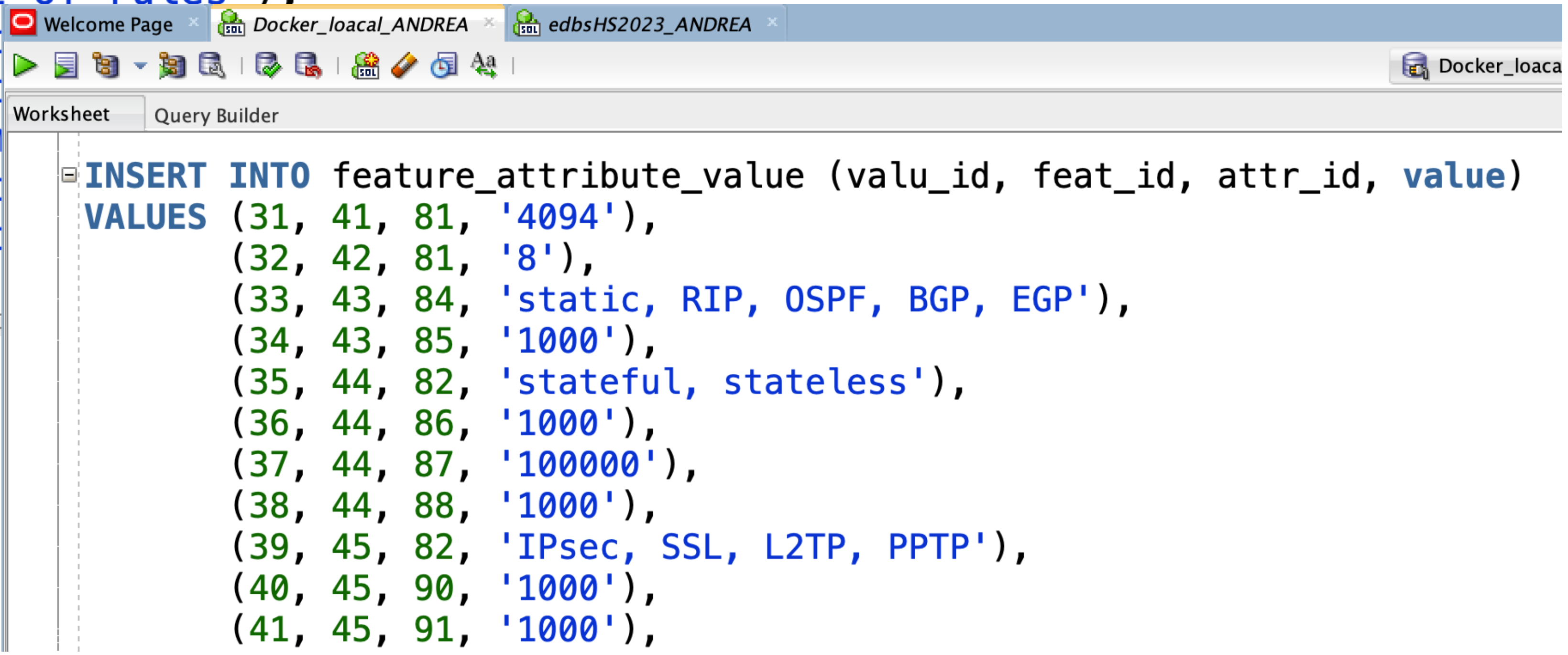
```
ALTER TABLE feature_eav ADD parent_feat_id NUMBER(10);
ALTER TABLE feature_eav ADD CONSTRAINT feae_feae_fk
  FOREIGN KEY (parent_feat_id) REFERENCES feature_eav (feat_id);
```

EAV

New data



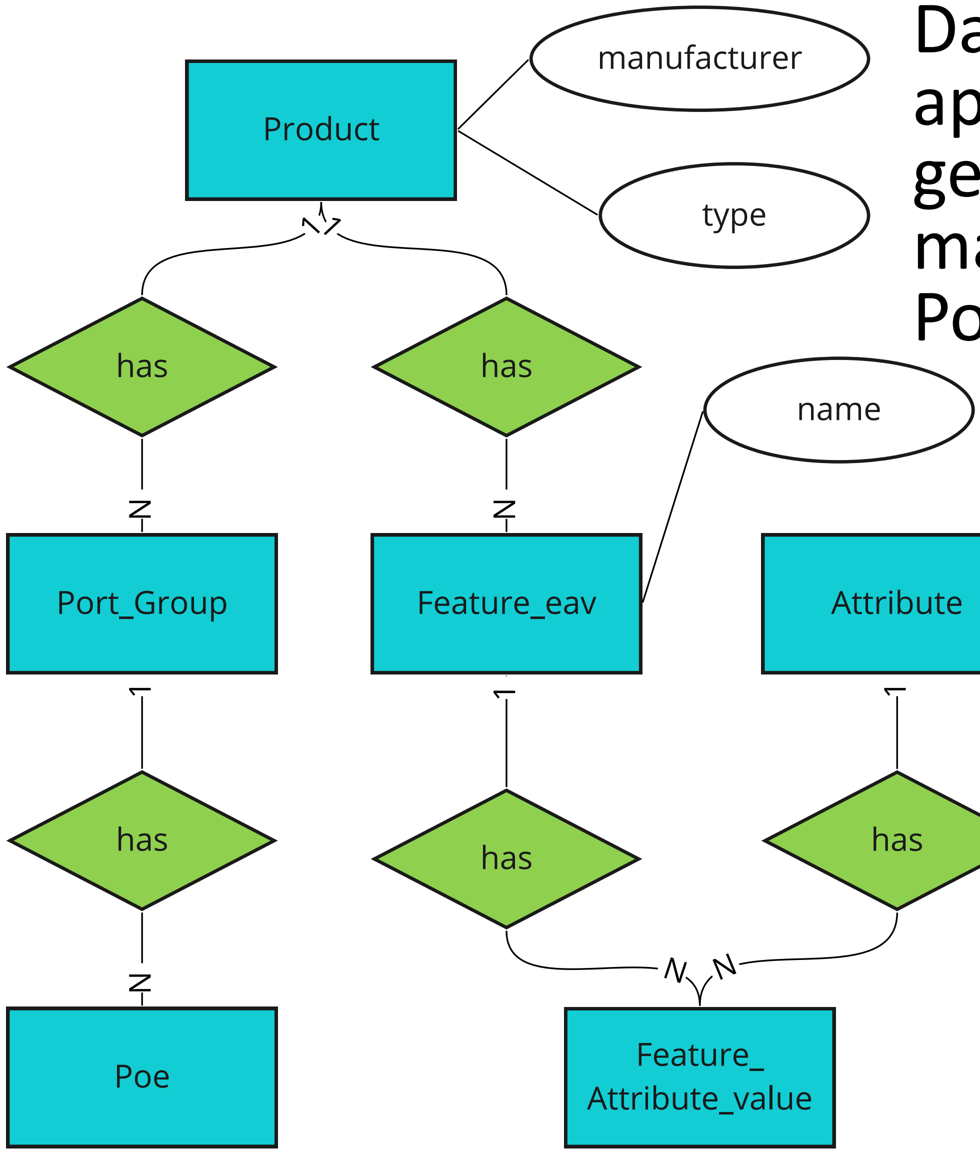
```
INSERT INTO feature_eav (feat_id, prod_id, name)...

INSERT INTO feature_eav (feat_id, parent_feat_id, name)...

INSERT INTO attribute (attr_id, name)
VALUES (86, 'amount_of_rules'),
       (87, 'amount
       (88, 'amount
       (89, 'ssl_in
       (90, 'amount
       (91, 'amount
```

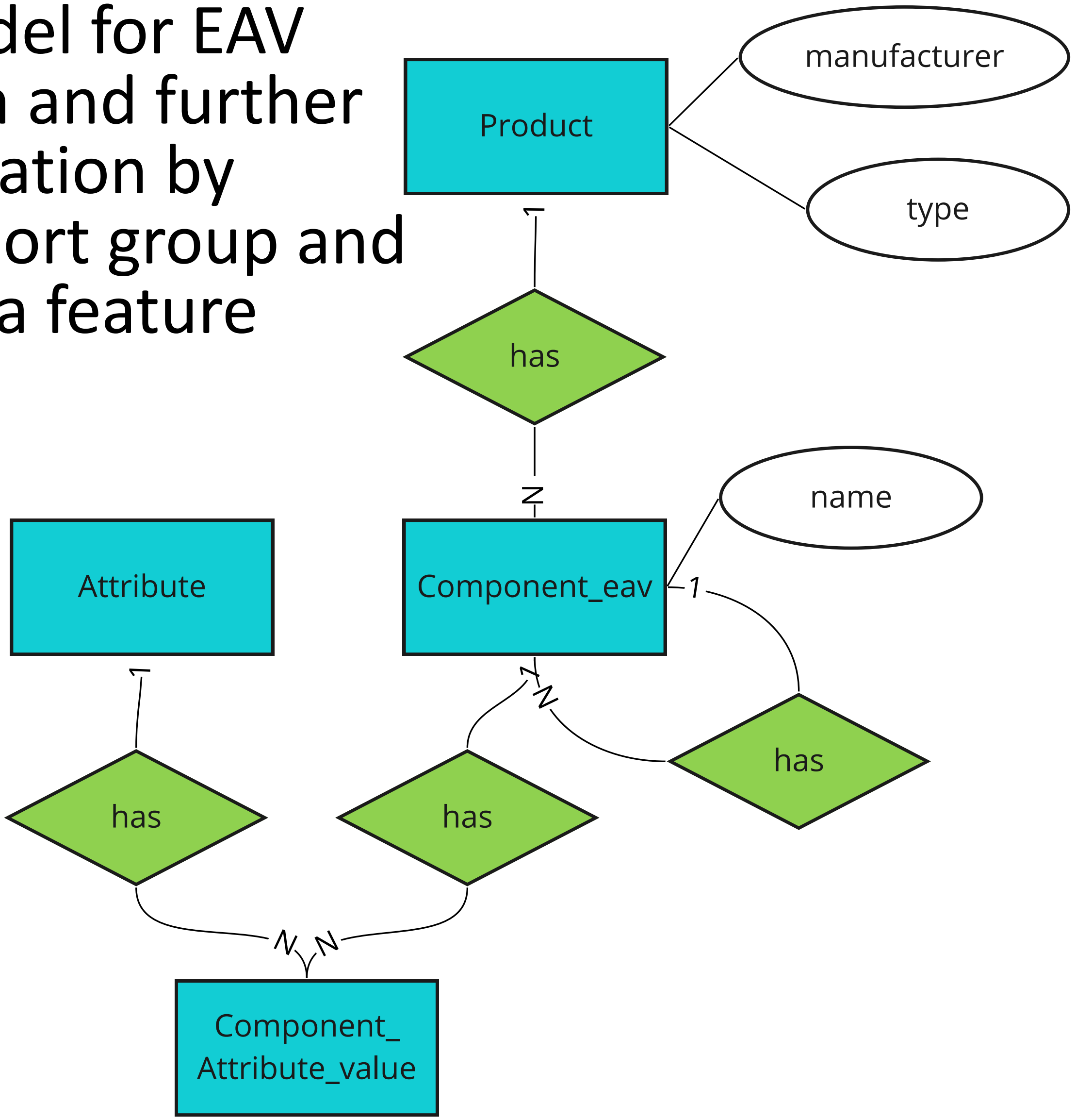


```
INSERT INTO feature_attribute_value (valu_id, feat_id, attr_id, value)
VALUES (31, 41, 81, '4094'),
       (32, 42, 81, '8'),
       (33, 43, 84, 'static, RIP, OSPF, BGP, EGP'),
       (34, 43, 85, '1000'),
       (35, 44, 82, 'stateful, stateless'),
       (36, 44, 86, '1000'),
       (37, 44, 87, '100000'),
       (38, 44, 88, '1000'),
       (39, 45, 82, 'IPsec, SSL, L2TP, PPTP'),
       (40, 45, 90, '1000'),
       (41, 45, 91, '1000'),
```

EAV

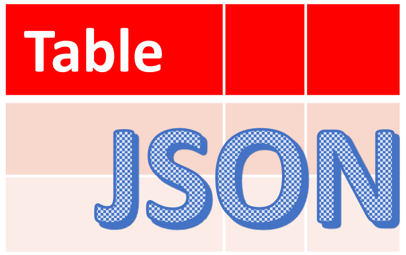
Data model for EAV approach and further generalisation by making port group and PoE also a feature





EAV

Would there be other solutions?

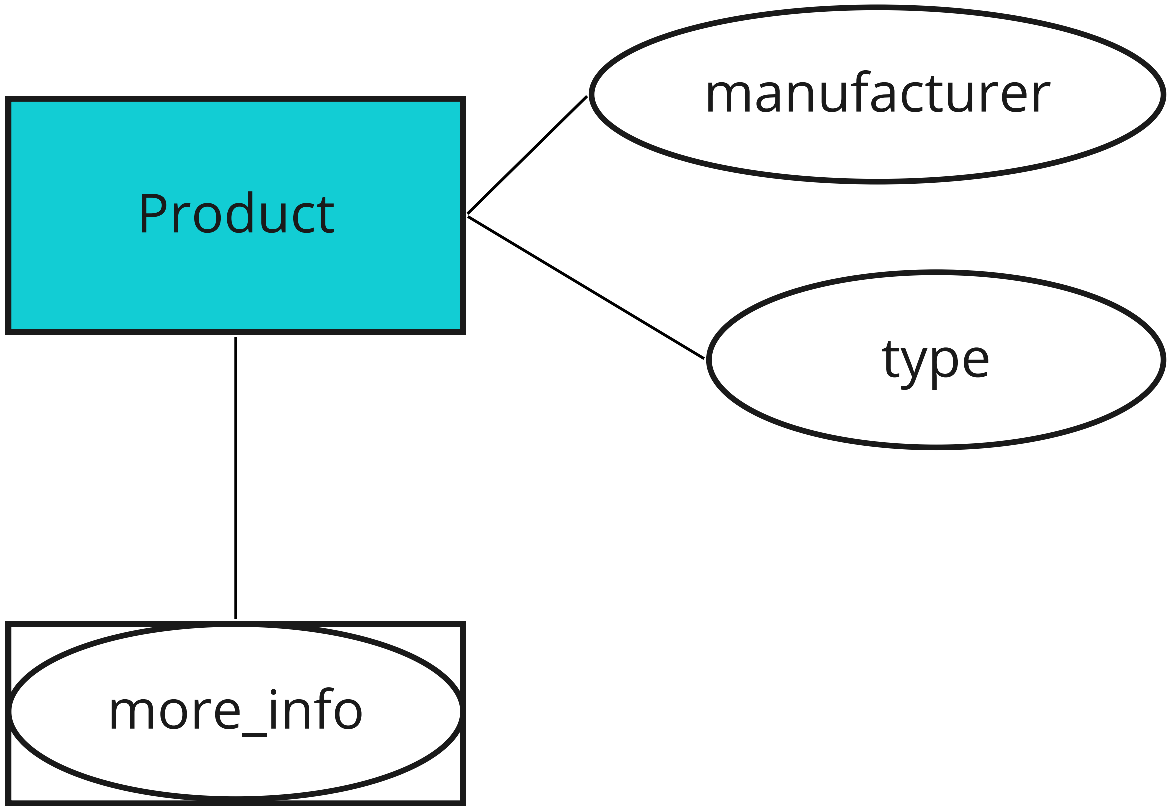


JSON in an attribute

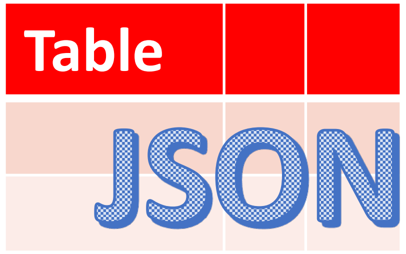
Explain principle

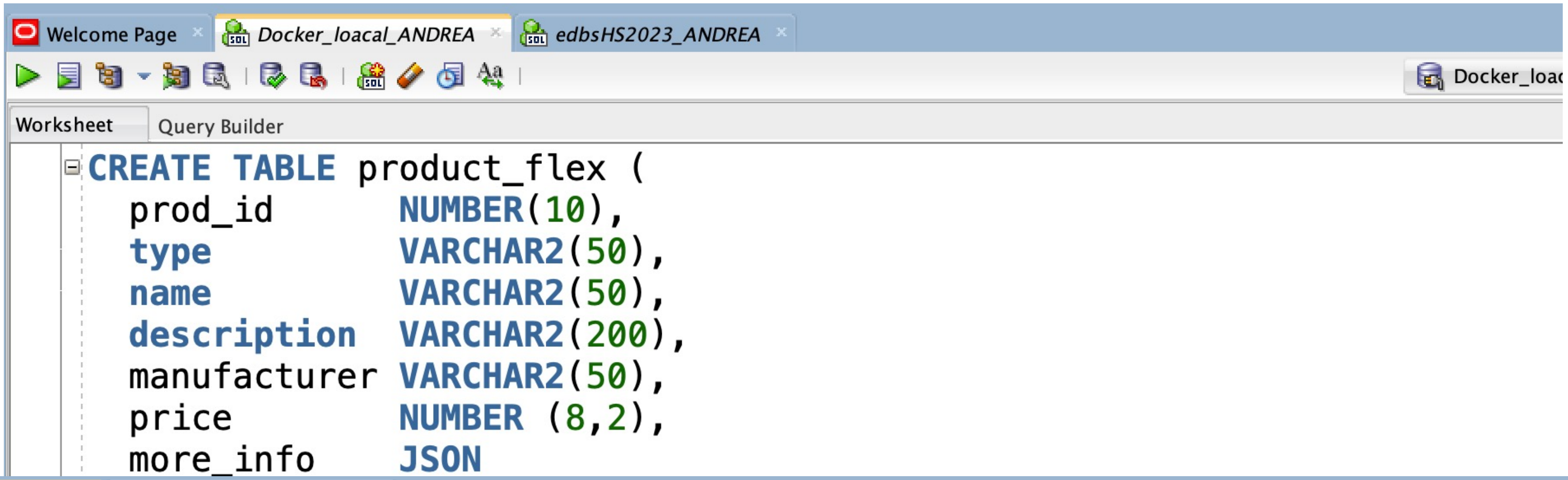
Data model

Demo

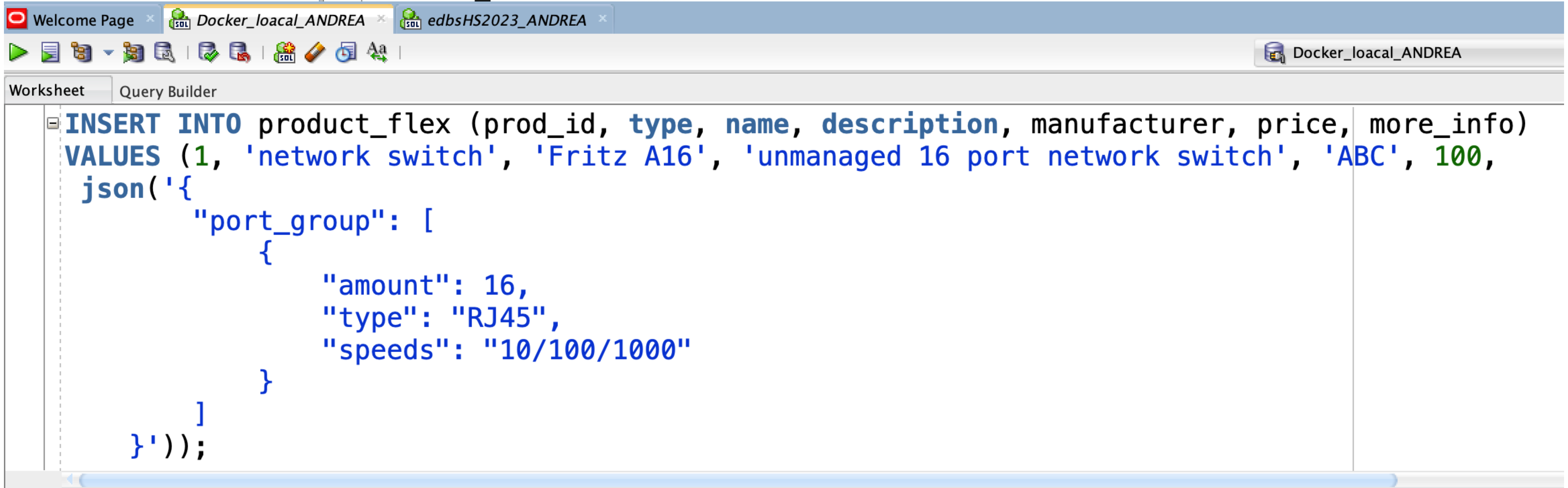


Would there be other solutions?





```
CREATE TABLE product_flex (
  prod_id      NUMBER(10),
  type         VARCHAR2(50),
  name         VARCHAR2(50),
  description  VARCHAR2(200),
  manufacturer VARCHAR2(50),
  price        NUMBER (8,2),
  more_info    JSON
```

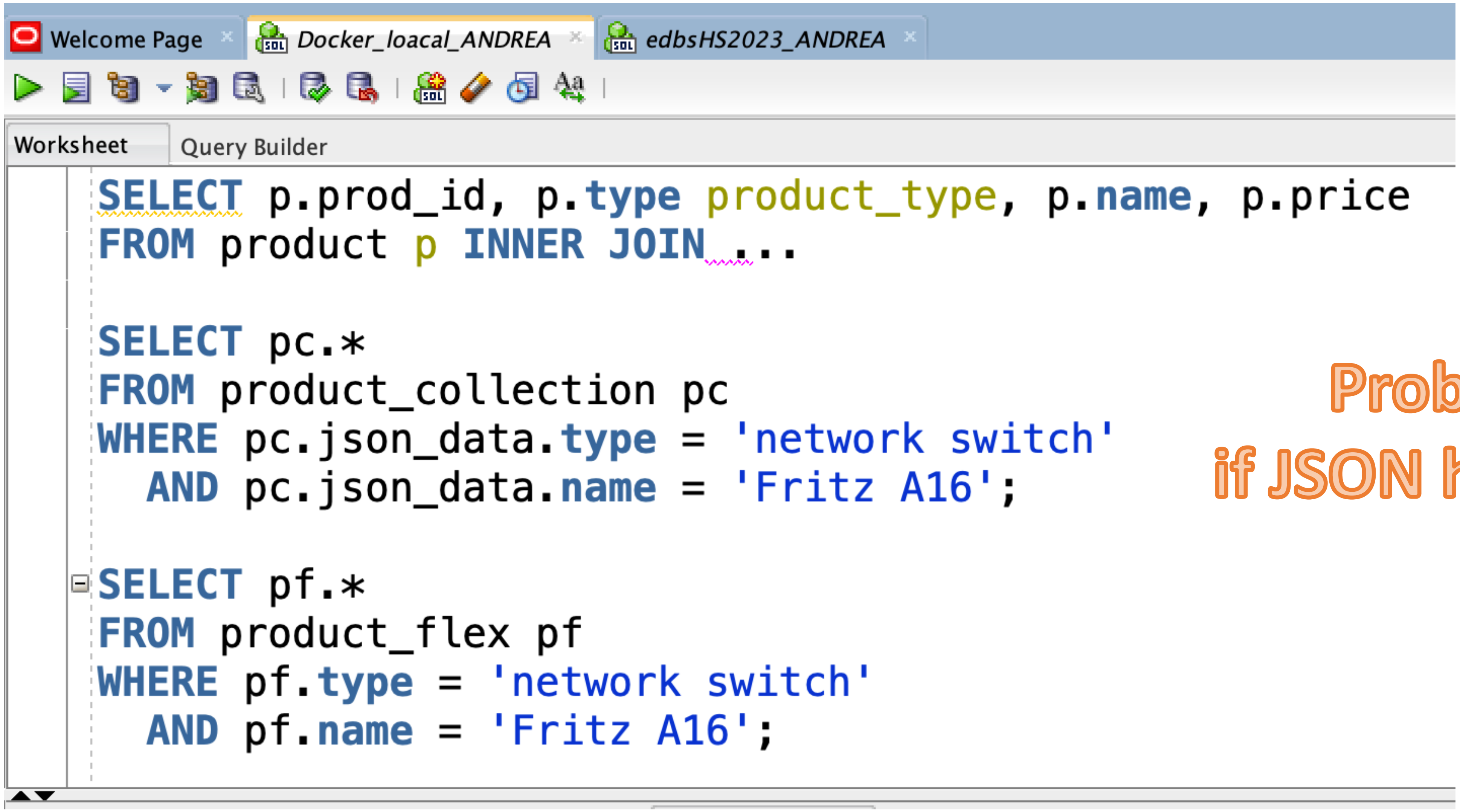


```
INSERT INTO product_flex (prod_id, type, name, description, manufacturer, price, more_info)
VALUES (1, 'network switch', 'Fritz A16', 'unmanaged 16 port network switch', 'ABC', 100,
 json('{
       "port_group": [
           {
               "amount": 16,
               "type": "RJ45",
               "speeds": "10/100/1000"
           }
       ]
    }'));
```

Do we have devices with a port group of type SFP?

```
SELECT pf.prod_id, pf.type product_type, pf.name,
    jt.*
FROM product_flex pf,
   JSON_TABLE(
     pf.more_info
     COLUMNS (
       NESTED port_group[*]
       COLUMNS (
         amount NUMBER(3) PATH '$.amount',
         type VARCHAR2(50) PATH '$.type'
       )
     )
   ) jt
WHERE jt.type = 'SFP';
```

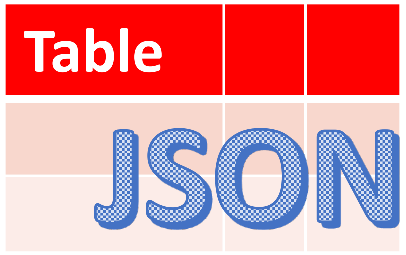
Give me all the details of the network switch “Fritz A16”



```
SELECT p.prod_id, p.type product_type, p.name, p.price
FROM product p INNER JOIN ...

SELECT pc.*
FROM product_collection pc
WHERE pc.json_data.type = 'network switch'
  AND pc.json_data.name = 'Fritz A16';

SELECT pf.*
FROM product_flex pf
WHERE pf.type = 'network switch'
  AND pf.name = 'Fritz A16';
```

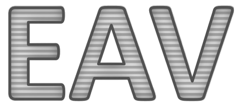
Problem
if JSON has typo

Summary and vote



- Table with JSON

- EAV



- Document Database

- Relationale Database

Quelle https://thenounproject.com/browse/icons/term/database-table/

Dr. Andrea Kennel

Consultant

Lecturer for Databases

Coach for Project Management

University of Applied Sciences Northwestern Switzerland

Brugg/Windisch, Switzerland

andrea.kennel@fhnw.ch
andrea@infokennel.ch
www.infokennel.ch