Hijerarhijski upiti

Boris Golub, dipl.ing.
Mihael Radovan, dipl.inf.
KONČAR - Inženjering za energetiku i transport d.d.

SADRŽAJ

I. UVOD

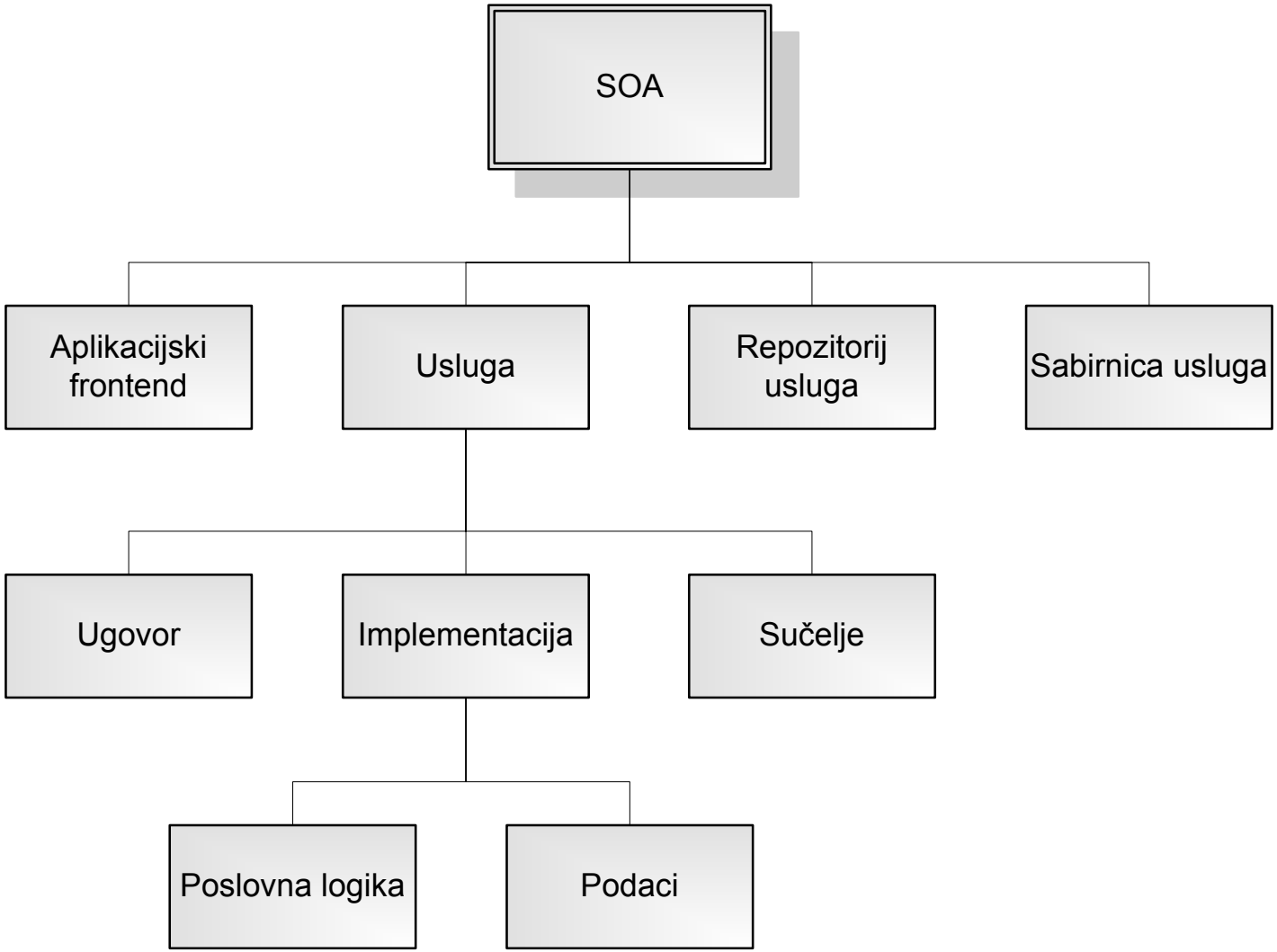
II. HIJERARHIJSKI UPITI

III. EXPLAIN PLAN TABLICA

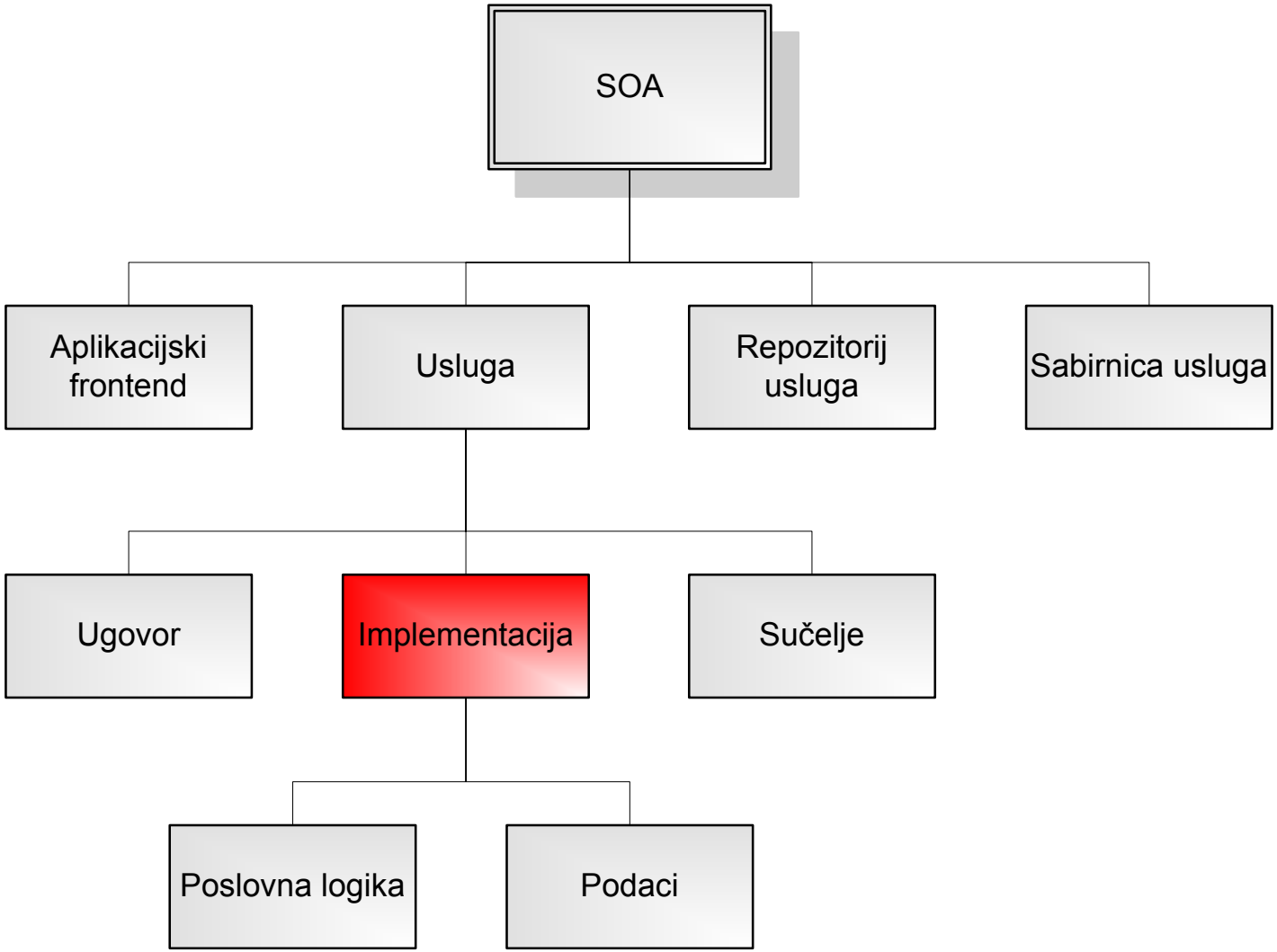
IV. COMMON INFORMATION MODEL

V. ZAKLJUČAK

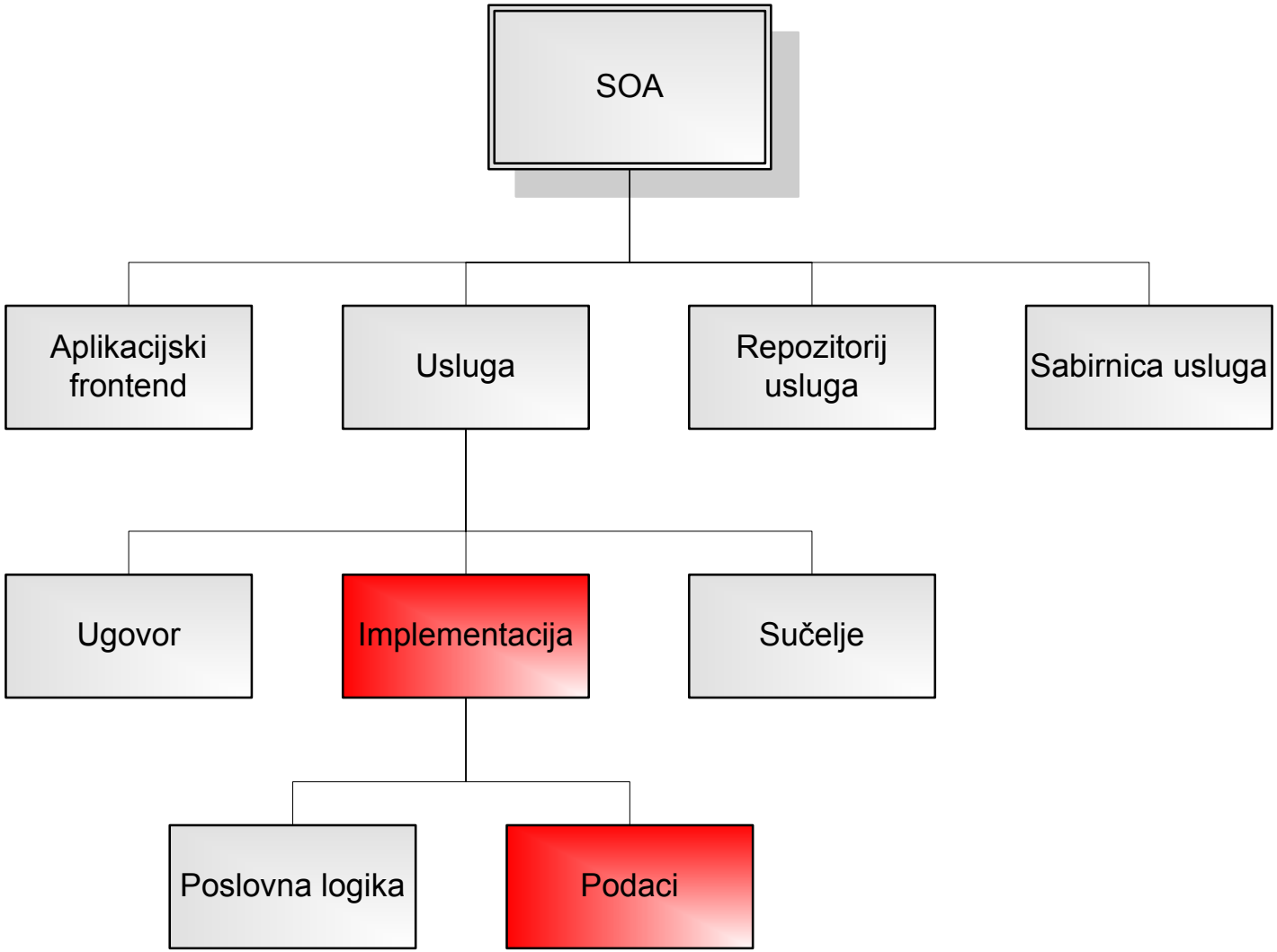
UVOD


SOA
Aplikacijski frontend
Usluga
Repozitorij usluga
Sabirnica usluga
Ugovor
Implementacija
Sučelje
Poslovna logika
Podaci

UVOD


SOA
Aplikacijski frontend
Usluga
Repozitorij usluga
Sabirnica usluga
Ugovor
Implementacija
Sučelje
Poslovna logika
Podaci

UVOD

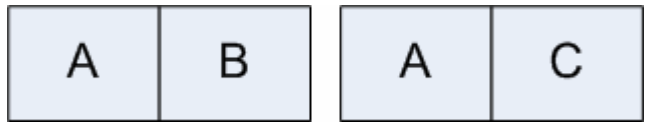

SOA
Aplikacijski frontend
Usluga
Repozitorij usluga
Sabirnica usluga
Ugovor
Implementacija
Sučelje
Poslovna logika
Podaci

Tipovi baza podataka

- Obična datoteka
- Hijerarhijska baza podataka
- Relacijska baza podataka
- Objektna baza podataka

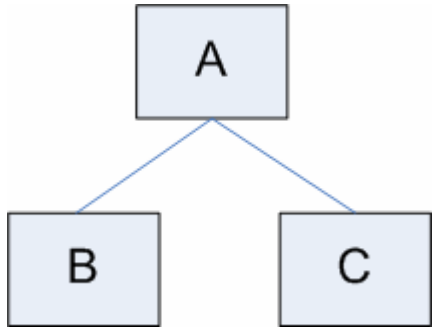
Tipovi baza podataka

- Obična datoteka
- Hijerarhijska baza podataka
- Relacijska baza podataka
- Objektna baza podataka



Tipovi baza podataka

- Obična datoteka
- Hijerarhijska baza podataka
- Relacijska baza podataka
- Objektna baza podataka



Tipovi baza podataka

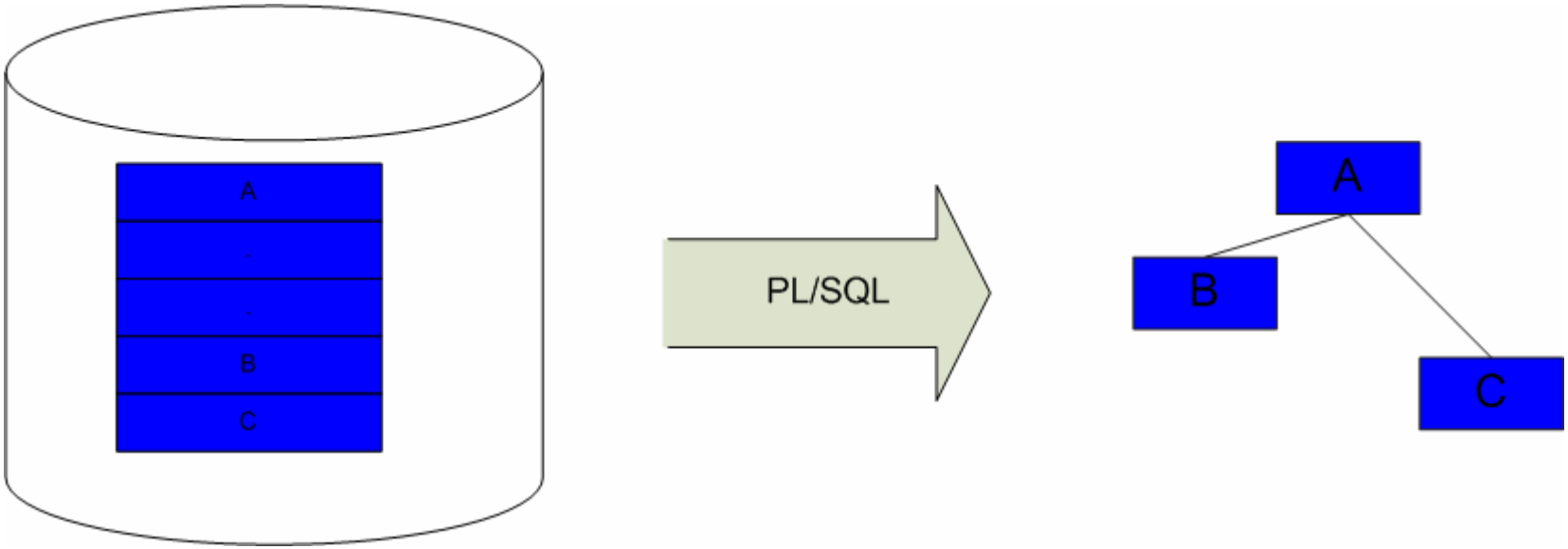
- Obična datoteka
- Hijerarhijska baza podataka
- Relacijska baza podataka
- Objektna baza podataka

A	B
A	C

Tipovi baza podataka

- Obična datoteka
- Hijerarhijska baza podataka
- Relacijska baza podataka
- Objektna baza podataka

Hijerarhijski upit



Relacijska baza podataka

Strukturirani podatak

SADRŽAJ

I. UVOD

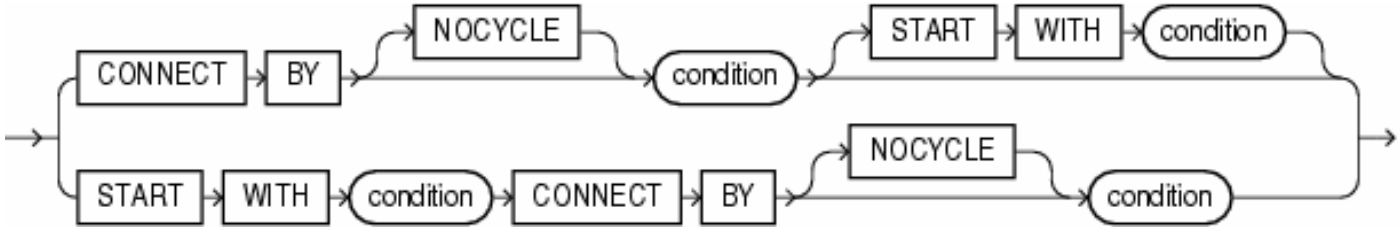
II. HIJERARHIJSKI UPITI

III. EXPLAIN PLAN TABLICA

IV. COMMON INFORMATION MODEL

V. ZAKLJUČAK

CONNECT BY ... START WITH ... PRIOR



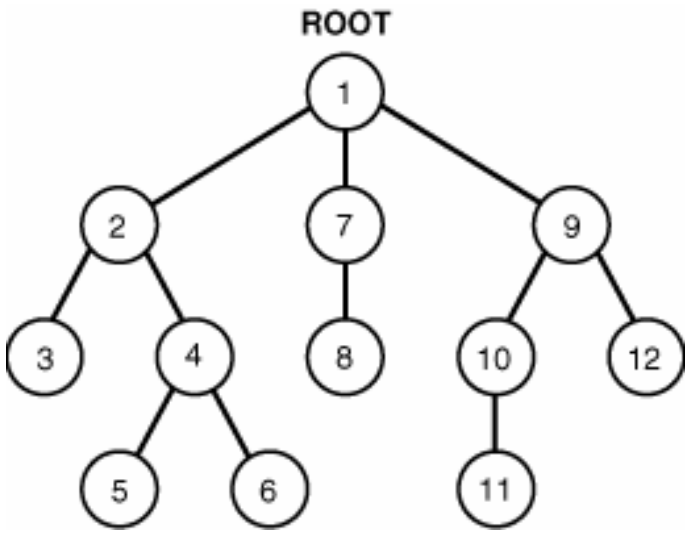

```
for rec in (select * from some_table) loop
  if FULLFILLS_START_WITH_CONDITION(rec) then
    RECURSE(rec, rec.child);
  end if;
end loop;

procedure RECURSE (rec in MATCHES_SELECT_STMT, new_parent IN field_type) is
  begin
  APPEND_RESULT_LIST(rec);
  for rec_recurse in (select * from some_table) loop
    if FULLFILLS_CONNECT_BY_CONDITION(rec_recurse.child, new_parent) then
      RECURSE(rec_recurse,rec_recurse.child);
    end if;
  end loop;
end procedure RECURSE;
```

Izvedba

Redoslijed izvođenja:

1. Prvo se izvode JOIN instrukcije (bilo iz FROM ili WHERE dijela)
2. Evaluira se CONNECT BY uvjet
3. Evaluira se preostali dio WHERE izraza



Opis izvođenja:

1. Selekcija onih redaka koji zadovoljavaju START WITH uvjet
2. Selektiraj djecu svakog korijenskog čvora
3. Prema CONNECT BY uvjetu selektira prvo djecu čvorova iz 2, a zatim to ponovlja sukcesivno
4. Svi preostali WHERE uvjet se izvršavaju pojedinačno na dobivenim recima (ne uklanjaju se apriori djeca čvora koji ih ne zadovoljava)
5. Reci se vraćaju u poretku prema slici – hijerarhijsko stablo

Mogućnosti

Operator:

- PRIOR
- *CONNECT_BY_ROOT*

Pseudokolone:

- LEVEL
- *CONNECT_BY_ISCYCLE*
- *CONNECT_BY_ISLEAF*

Funkcije:

- *SYS_CONNECT_BY_PATH*

Ključne riječi:

- *NOCYCLE*

Redoslijed prikaza:

- ORDER SIBLINGS BY

PRIOR operator:

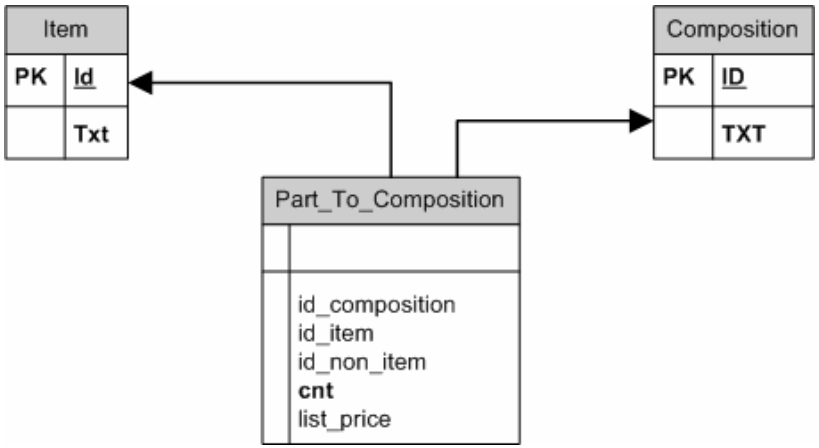
- Jedan od izraza u CONNECT BY mora biti označen PRIOR operatorom kako bi naznačio roditelja
- U višestrukim izrazima u uvjetu dovoljno je navesti ga samo jednom, ali može i više puta
- Unarni operator istog prioriteta kao unarni + i - , evaluira neposredni sljedni izraz na vrijednost roditelja trenutnog retka
- Uglavnom se koristi uz operator jednakosti , ali može i uz druge operatore uz mogućnost beskonačne petlje

Jednostavan primjer

```
create table item (
  id   number primary key,
  txt  varchar2(25) not null
);

create table composition (
  id  number primary key,
  txt varchar2(25) not null
);

create table part_to_composition (
  id_composition references composition not null,
  id_item       references item,
  id_non_item   references composition,
  cnt           number(5)  not null check (cnt > 0),
  list_price    number,
  check (id_item is null and id_non_item is not null or id_item is not null and id_non_item is null)
);
```

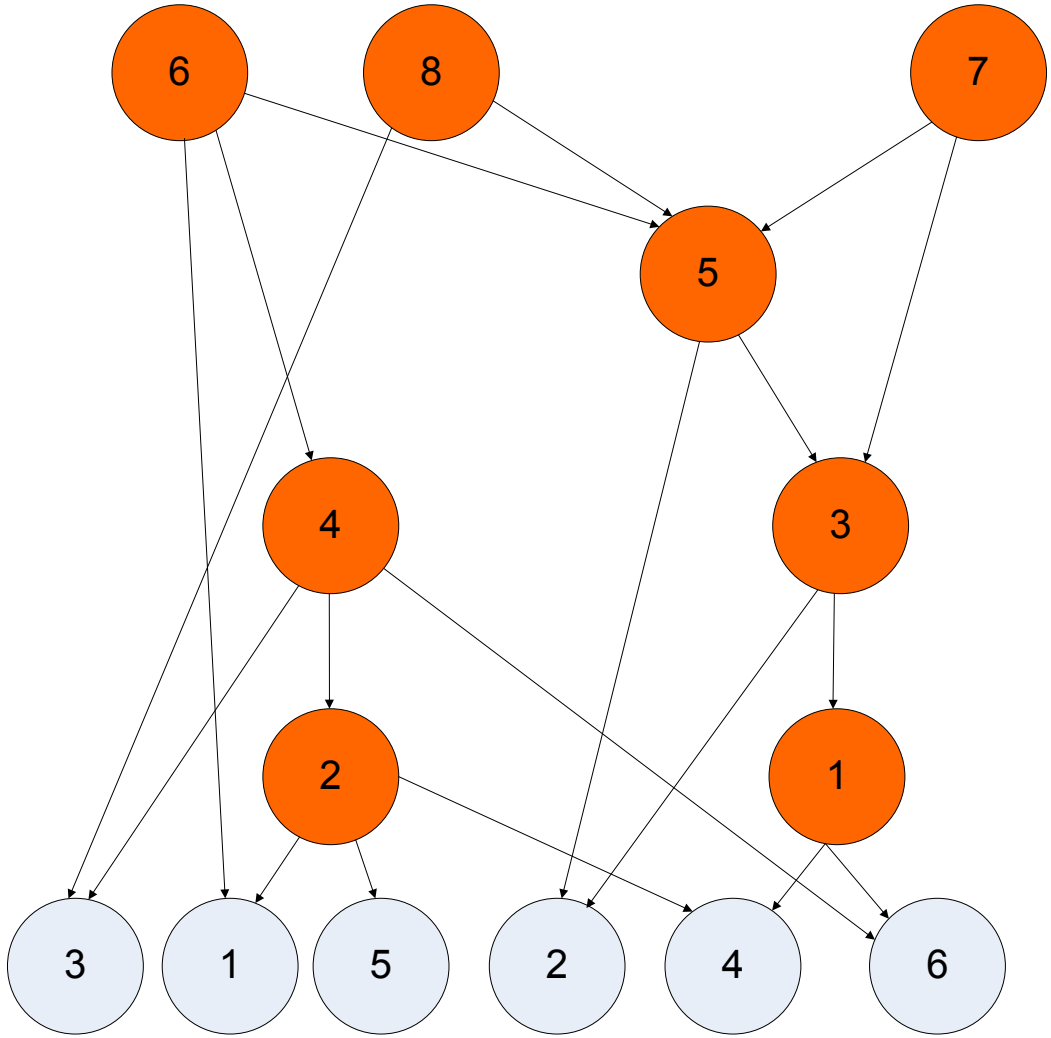


Row #	ID	TXT
1	1	item 1
2	2	item 2
3	3	item 3
4	4	item 4
5	5	item 5
6	6	item 6

Row #	ID	TXT
1	1	comp 1
2	2	comp 2
3	3	comp 3
4	4	comp 4
5	5	comp 5
6	6	prod 1
7	7	prod 2
8	8	prod 3

Row #	ID_COMPOSITION	ID_ITEM	ID_NON_ITEM	CNT	LIST_PRICE
1	1	4		2	0
2	1	6		3	0
3	2	1		5	0
4	2	4		1	0
5	2	5		3	0
6	3	2		2	0
7	3		1	2	0
8	4		2	1	0
9	4	3		4	0
10	4	6		1	0
11	5		3	1	0
12	5	2		4	0
13	6		5	1	0
14	6		4	1	0
15	6	1		1	0
16	7		5	1	0
17	7		3	2	0
18	8		5	1	0
19	8	3		2	0

Hijerarhijski graf



SYS_CONNECT_BY_PATH (column, char)

- Daje put od korijena do konkretnog djeteta
- Validan samo u hijerarhijskim upitima
- Putanja je razdvojena proizvoljnim znakom

```
SELECT     LEVEL,
           LPAD (' ', 2 * LEVEL - 1)
        || SYS_CONNECT_BY_PATH (NVL (composition.txt, item.txt), '/') path,
        ' [' || cnt || 'x' || '] ' cnt
    FROM part_to_composition LEFT JOIN item ON part_to_composition.id_item =
                                                item.ID
         LEFT JOIN composition ON part_to_composition.id_non_item =
                                                composition.ID
START WITH id_composition = 7
CONNECT BY id_composition = PRIOR id_non_item;
```

Row #	LEVEL	PATH	CNT
1	1	/comp 3	[2x]
2	2	/comp 3/comp 1	[2x]
3	3	/comp 3/comp 1/item 6	[3x]
4	3	/comp 3/comp 1/item 4	[2x]
5	2	/comp 3/item 2	[2x]
6	1	/comp 5	[1x]
7	2	/comp 5/comp 3	[1x]
8	3	/comp 5/comp 3/comp 1	[2x]
9	4	/comp 5/comp 3/comp 1/item 6	[3x]
10	4	/comp 5/comp 3/comp 1/item 4	[2x]
11	3	/comp 5/comp 3/item 2	[2x]
12	2	/comp 5/item 2	[4x]

CONNECT BY ROOT

- Unarni operator
- Za zadano polje vraća korijensku vrijednost

```
SELECT     LEVEL,
           LPAD (' ', 2 * LEVEL - 1)
        || SYS_CONNECT_BY_PATH (NVL (composition.txt, item.txt), '/') path,
        CONNECT_BY_ROOT NVL(composition.txt, item.txt) korijen,
        ' [' || cnt || 'x' || '] ' cnt
    FROM part_to_composition LEFT JOIN item ON part_to_composition.id_item =
                                                item.ID
        LEFT JOIN composition ON part_to_composition.id_non_item =
                                                composition.ID
START WITH id_composition = 7
CONNECT BY id_composition = PRIOR id_non_item;
```

Row #	LEVEL	PATH	KORIJEN	CNT
1	1	/comp 3	comp 3	[2x]
2	2	/comp 3/comp 1	comp 3	[2x]
3	3	/comp 3/comp 1/item 6	comp 3	[3x]
4	3	/comp 3/comp 1/item 4	comp 3	[2x]
5	2	/comp 3/item 2	comp 3	[2x]
6	1	/comp 5	comp 5	[1x]
7	2	/comp 5/comp 3	comp 5	[1x]
8	3	/comp 5/comp 3/comp 1	comp 5	[2x]
9	4	/comp 5/comp 3/comp 1/item 6	comp 5	[3x]
10	4	/comp 5/comp 3/comp 1/item 4	comp 5	[2x]
11	3	/comp 5/comp 3/item 2	comp 5	[2x]
12	2	/comp 5/item 2	comp 5	[4x]

CONNECT_BY_ISLEAF

- Pseudokolona koja vraća 1 ukoliko je trenutni redak list stabla definiranog hijerarhijskim upitom

```
SELECT     LEVEL,
           LPAD (' ', 2 * LEVEL - 1)
        || SYS_CONNECT_BY_PATH (NVL (composition.txt, item.txt), '/') path,
        CONNECT_BY_ROOT NVL(composition.txt, item.txt) korijen, connect_by_isleaf,
        ' [' || cnt || 'x' || '] ' cnt
    FROM part_to_composition LEFT JOIN item ON part_to_composition.id_item =
                                                item.ID
        LEFT JOIN composition ON part_to_composition.id_non_item =
                                                composition.ID
START WITH id_composition = 7
CONNECT BY id_composition = PRIOR id_non_item;
```

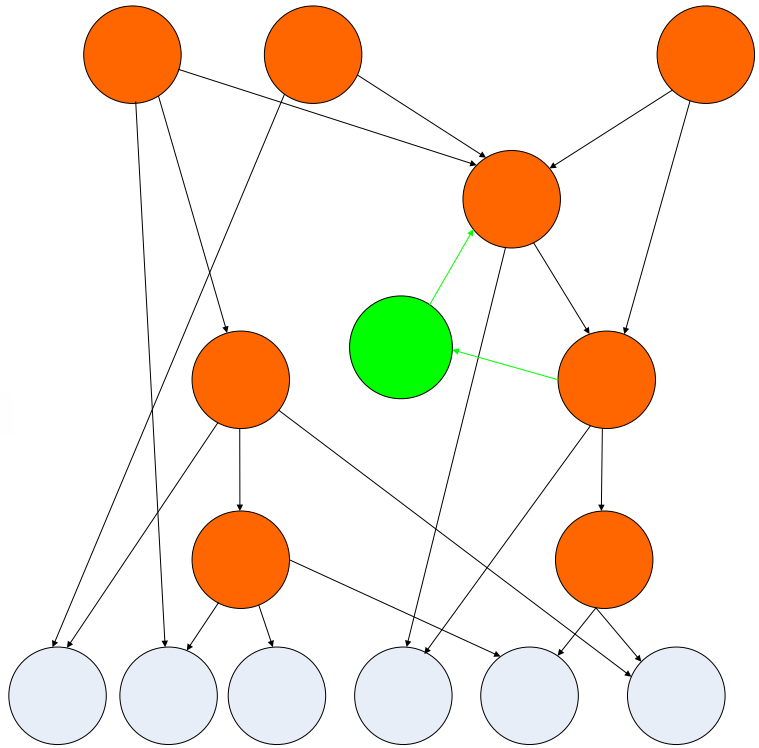
Row #	LEVEL	PATH	KORIJEN	CONNECT_BY_ISLEAF	CNT
1	1	/comp 3	comp 3	0	[2x]
2	2	/comp 3/comp 1	comp 3	0	[2x]
3	3	/comp 3/comp 1/item 6	comp 3	1	[3x]
4	3	/comp 3/comp 1/item 4	comp 3	1	[2x]
5	2	/comp 3/item 2	comp 3	1	[2x]
6	1	/comp 5	comp 5	0	[1x]
7	2	/comp 5/comp 3	comp 5	0	[1x]
8	3	/comp 5/comp 3/comp 1	comp 5	0	[2x]
9	4	/comp 5/comp 3/comp 1/item 6	comp 5	1	[3x]
10	4	/comp 5/comp 3/comp 1/item 4	comp 5	1	[2x]
11	3	/comp 5/comp 3/item 2	comp 5	1	[2x]
12	2	/comp 5/item 2	comp 5	1	[4x]

CONNECT_BY_ISCYCLE

- Pseudokolona koja vraća 1 ukoliko trenutni redak ima djece koja su ujedno i njegovi preci, tj. dio je ciklusa

```
SELECT     LEVEL,
           LPAD (' ', 2 * LEVEL - 1)
        || SYS_CONNECT_BY_PATH (NVL (composition.txt, item.txt), '/') path,
        CONNECT_BY_ROOT NVL(composition.txt, item.txt) korijen, connect_by_isleaf,
        ' [' || cnt || 'x' || '] ' cnt
    FROM part_to_composition LEFT JOIN item ON part_to_composition.id_item =
                                                item.ID
        LEFT JOIN composition ON part_to_composition.id_non_item =
                                              composition.ID
START WITH id_composition = 7
CONNECT BY id_composition = PRIOR id_non_item;
```

[1]: (Error): ORA-01436: CONNECT BY loop in user data



```
SELECT     LEVEL,
           LPAD (' ', 2 * LEVEL - 1)
        || SYS_CONNECT_BY_PATH (NVL (composition.txt, item.txt), '/') path,
        CONNECT_BY_ROOT NVL(composition.txt, item.txt) korijen, connect_by_iscycle,
        ' [' || cnt || 'x' || '] ' cnt
    FROM part_to_composition LEFT JOIN item ON part_to_composition.id_item =
                                                item.ID
        LEFT JOIN composition ON part_to_composition.id_non_item =
                                              composition.ID
                                              where connect_by_iscycle=1
START WITH id_composition = 7
CONNECT BY NOCYCLE id_composition = PRIOR id_non_item;
```

Row #	LEVEL	PATH	KORIJEN	CONNECT_BY_ISCYCLE	CNT
1	3	/comp 3/za ciklus/comp 5	comp 3	1	[2x]
2	3	/comp 5/comp 3/za ciklus	comp 5	1	[2x]

ORDER SIBLINGS BY

- Čuva poredak strukture
- Sortira djecu na istom nivou prema danom kriteriju

```
SELECT     LEVEL,
           LPAD (' ', 2 * LEVEL - 1)
        || SYS_CONNECT_BY_PATH (NVL (composition.txt, item.txt), '/') path,
        CONNECT_BY_ROOT NVL(composition.txt, item.txt) korijen, connect_by_iscycle,
        ' [' || cnt || 'x' || '] ' cnt
    FROM part_to_composition LEFT JOIN item ON part_to_composition.id_item =
                                                item.ID
        LEFT JOIN composition ON part_to_composition.id_non_item =
                                                composition.ID

START WITH id_composition = 7
CONNECT BY nocycle id_composition = PRIOR id_non_item
order siblings by NVL(composition.txt, item.txt);
```

Row #	LEVEL	PATH	KORIJEN	CONNECT_BY_ISCYCLE	CNT
1	1	/comp 3	comp 3	0	[2x]
2	2	/comp 3/comp 1	comp 3	0	[2x]
3	3	/comp 3/comp 1/item 4	comp 3	0	[2x]
4	3	/comp 3/comp 1/item 6	comp 3	0	[3x]
5	2	/comp 3/item 2	comp 3	0	[2x]
6	2	/comp 3/za ciklus	comp 3	0	[2x]
7	3	/comp 3/za ciklus/comp 5	comp 3	1	[2x]
8	4	/comp 3/za ciklus/comp 5/item 2	comp 3	0	[4x]
9	1	/comp 5	comp 5	0	[1x]
10	2	/comp 5/comp 3	comp 5	0	[1x]
11	3	/comp 5/comp 3/comp 1	comp 5	0	[2x]
12	4	/comp 5/comp 3/comp 1/item 4	comp 5	0	[2x]
13	4	/comp 5/comp 3/comp 1/item 6	comp 5	0	[3x]
14	3	/comp 5/comp 3/item 2	comp 5	0	[2x]
15	3	/comp 5/comp 3/za ciklus	comp 5	1	[2x]
16	2	/comp 5/item 2	comp 5	0	[4x]

SADRŽAJ

I. UVOD

II. HIJERARHIJSKI UPITI

III. EXPLAIN PLAN TABLICA

IV. COMMON INFORMATION MODEL

V. ZAKLJUČAK

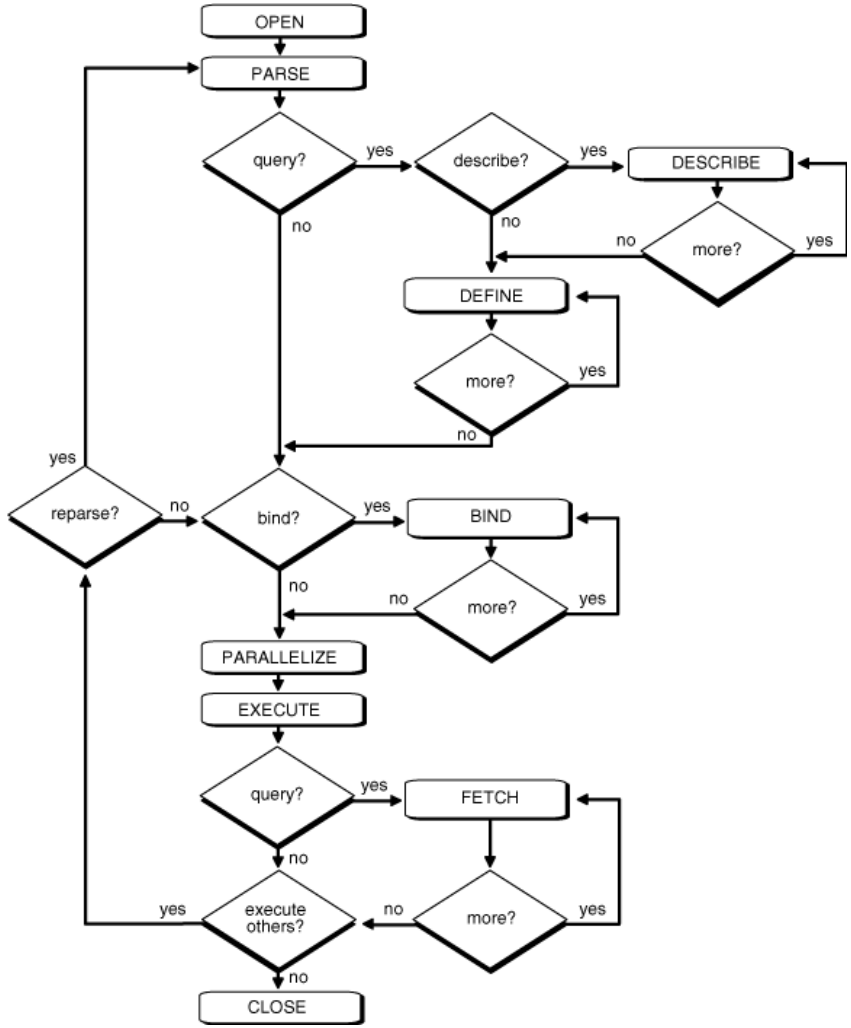
Explain plan tablica

```
explain plan <into table_name> for neki_sql_upit;
```

```
explain plan for select
  level, substr(lpad(' ', 2*level-1) ||
    nvl(composition.txt, item.txt), 1, 40) txt,
    ' [' || cnt || 'x' || '] ' cnt
from part_to_composition left join item        on
part_to_composition.id_item     = item.id
                    left join composition on
part_to_composition.id_non_item = composition.id
start with id_composition = 7
connect by id_composition = prior id_non_item;
```

```
select
  substr (lpad(' ', level-1) || operation || ' (' || options ||
')',1,30 ) "Operation",
  object_name
"Object"
from
  plan_table
start with id = 0
connect by prior id=parent_id;
```

```
select plan_table_output from
table(dbms_xplan.display('my_plan_table',null,'serial'))
;
```



SADRŽAJ

I. UVOD

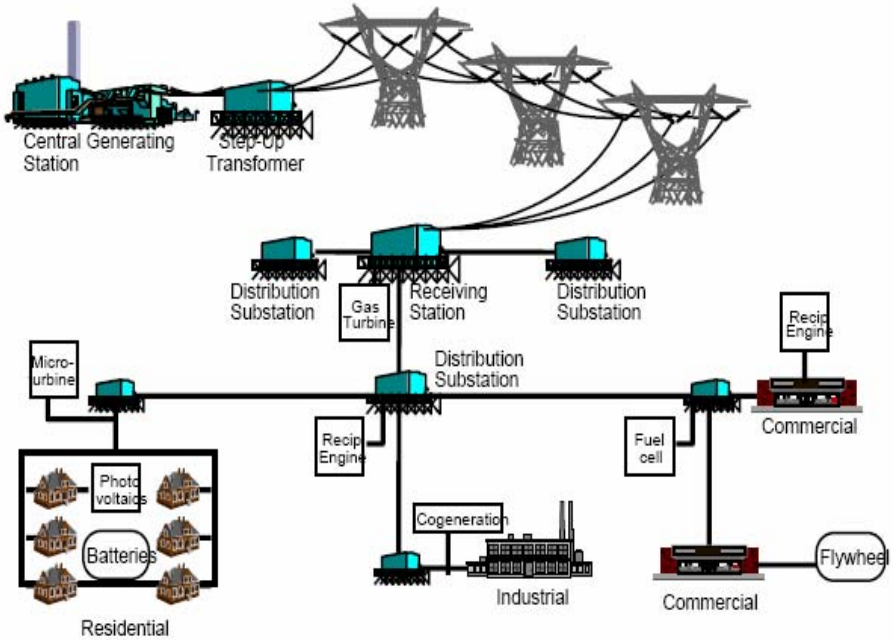
II. HIJERARHIJSKI UPITI

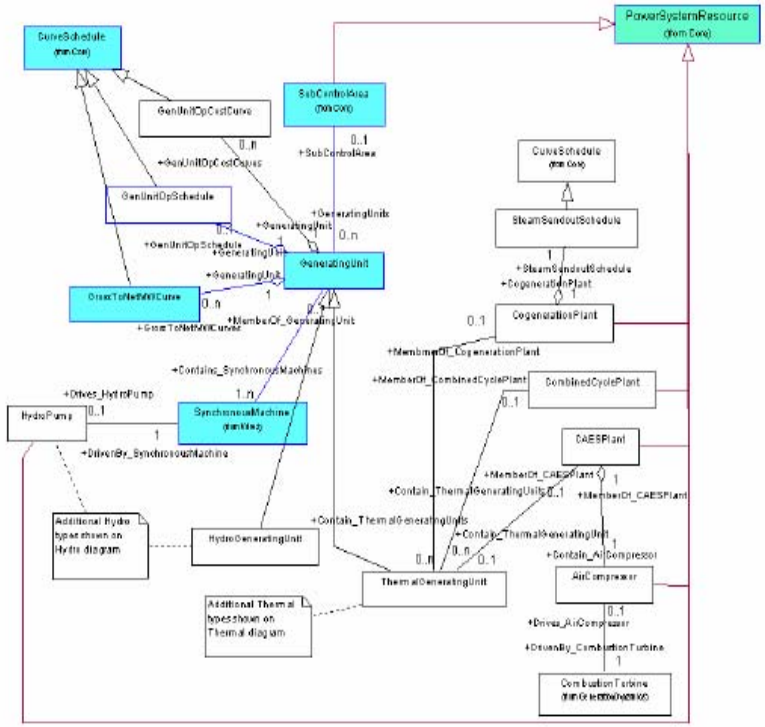
III. EXPLAIN PLAN TABLICA

IV. COMMON INFORMATION MODEL

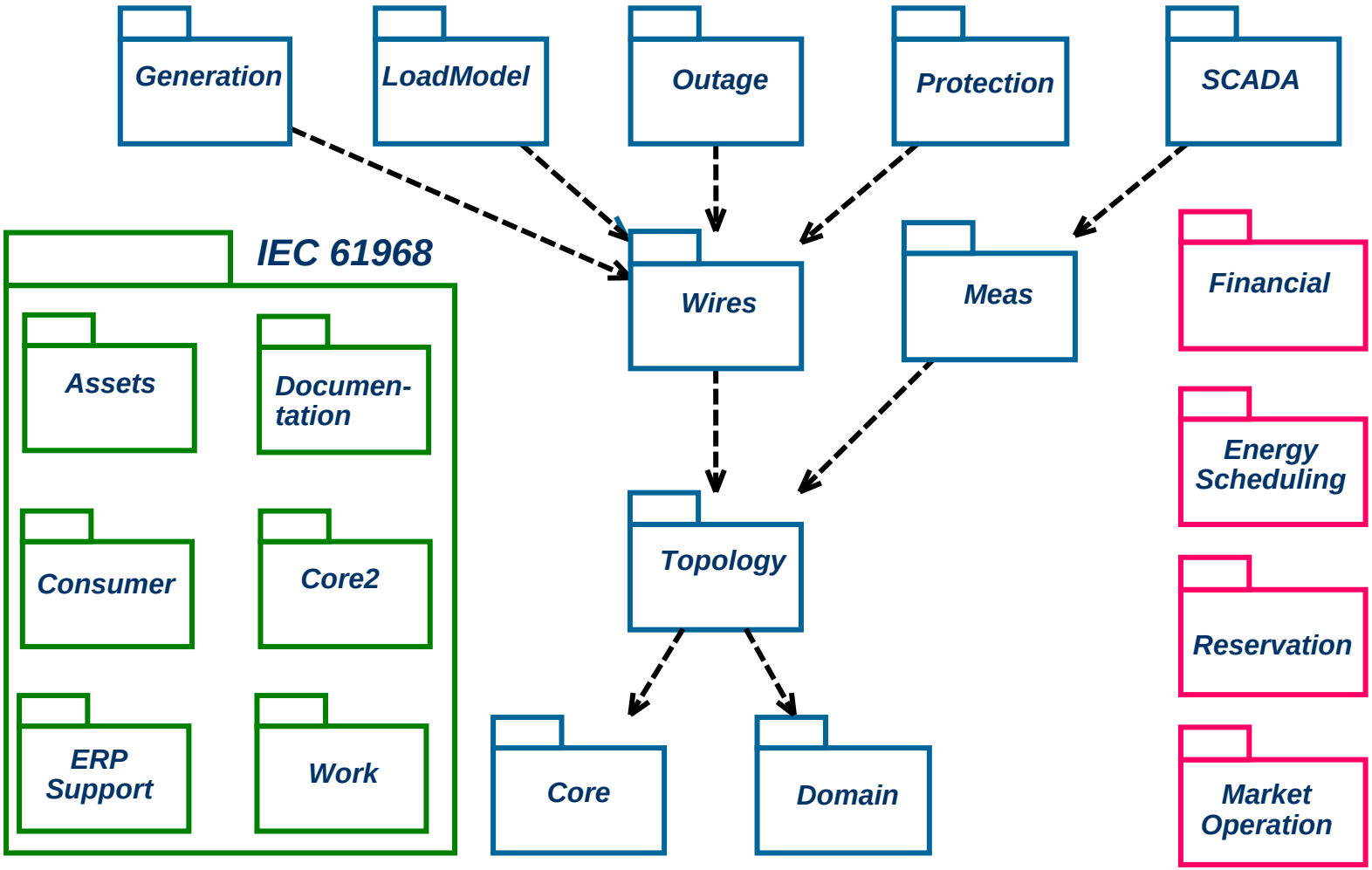
V. ZAKLJUČAK

CIM (Common Information Model)

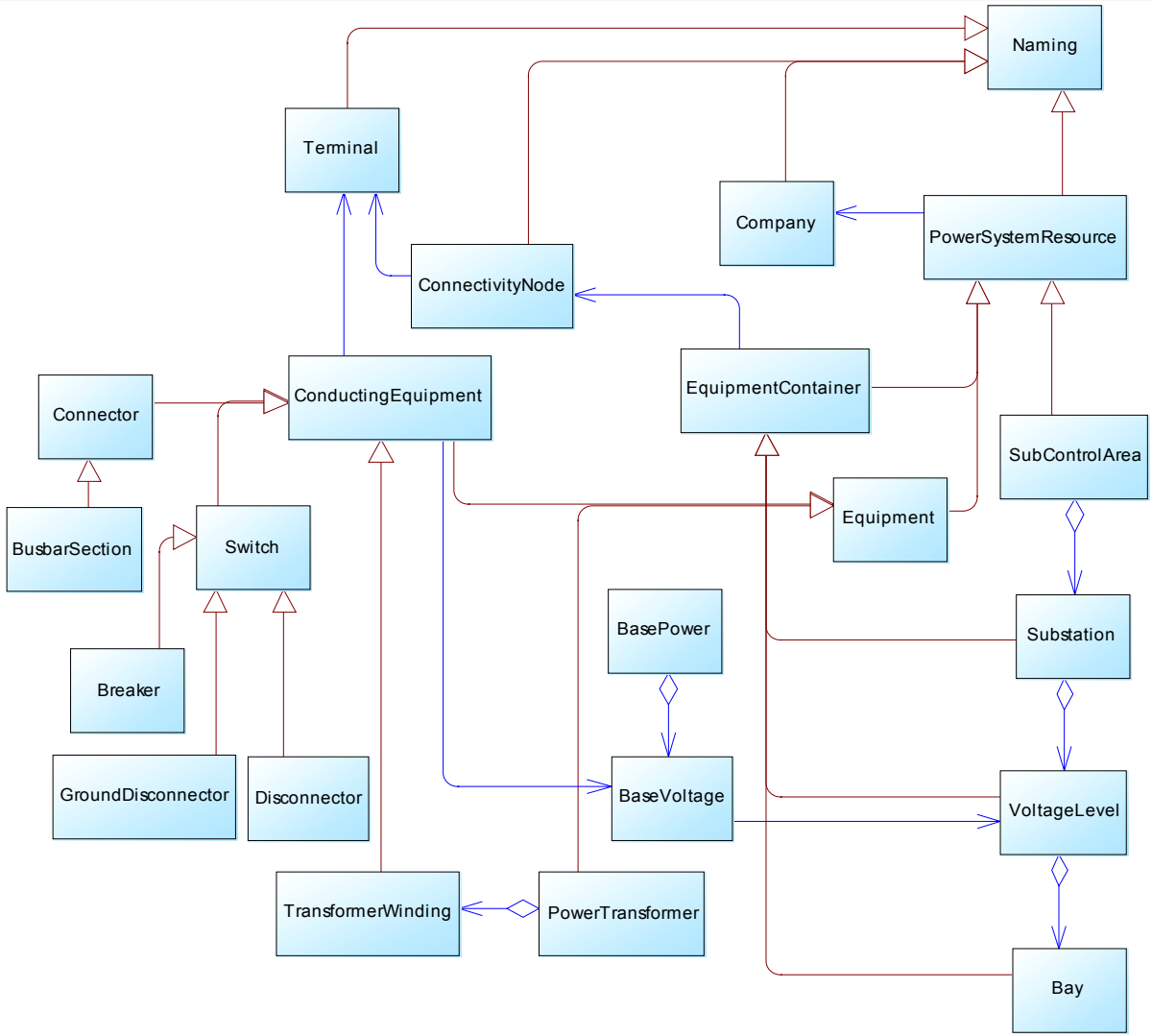




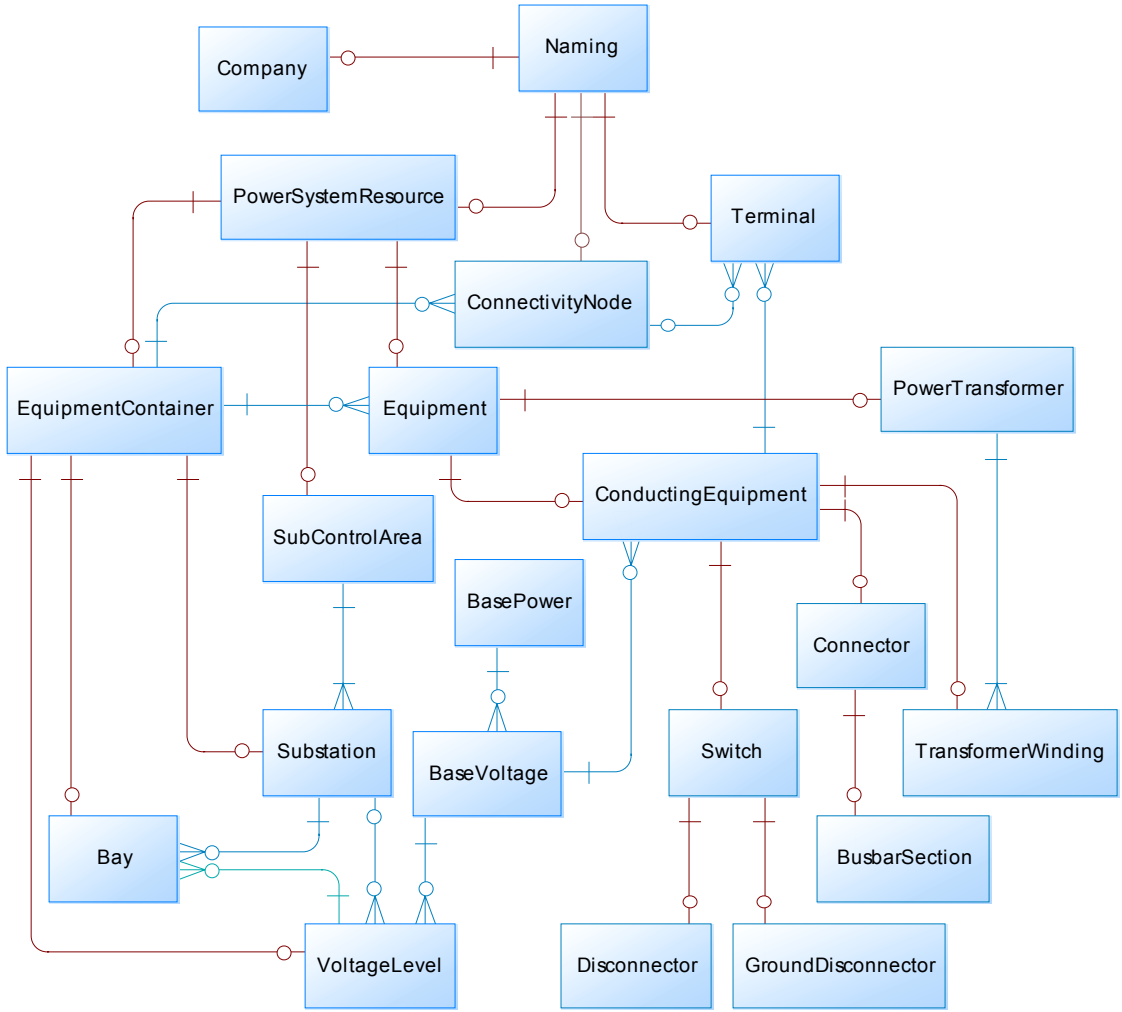
Struktura CIM modela



CIM objektni model

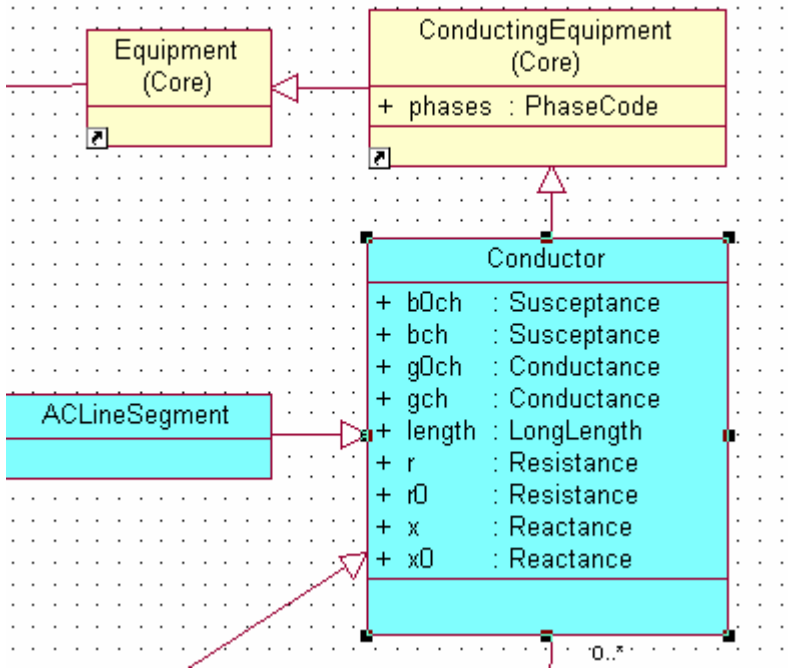


CIM relacijski model



Mapiranje objektni model -> relacijski model

Objektni model:



Relacijski model:

ID	CONDUCTORTYPE_ID	B0CH	BCH	G0CH	GCH	LENGTH	R	R0	X	X0
_16eb8d2d717f484			87,5998				3,3919559		14,292712	
_291e436d5cc41ac			105,5998				1,399236		4,2713518	
_40249b00be2b4a:			0				0,76955998		2,0448999	
_107ea0f45ac74aa			0				3,749548		4,2277403	

Modeli mapiranja

- *Jedna tablica sa svim atributima (vlastitim i naslijeđenim)*

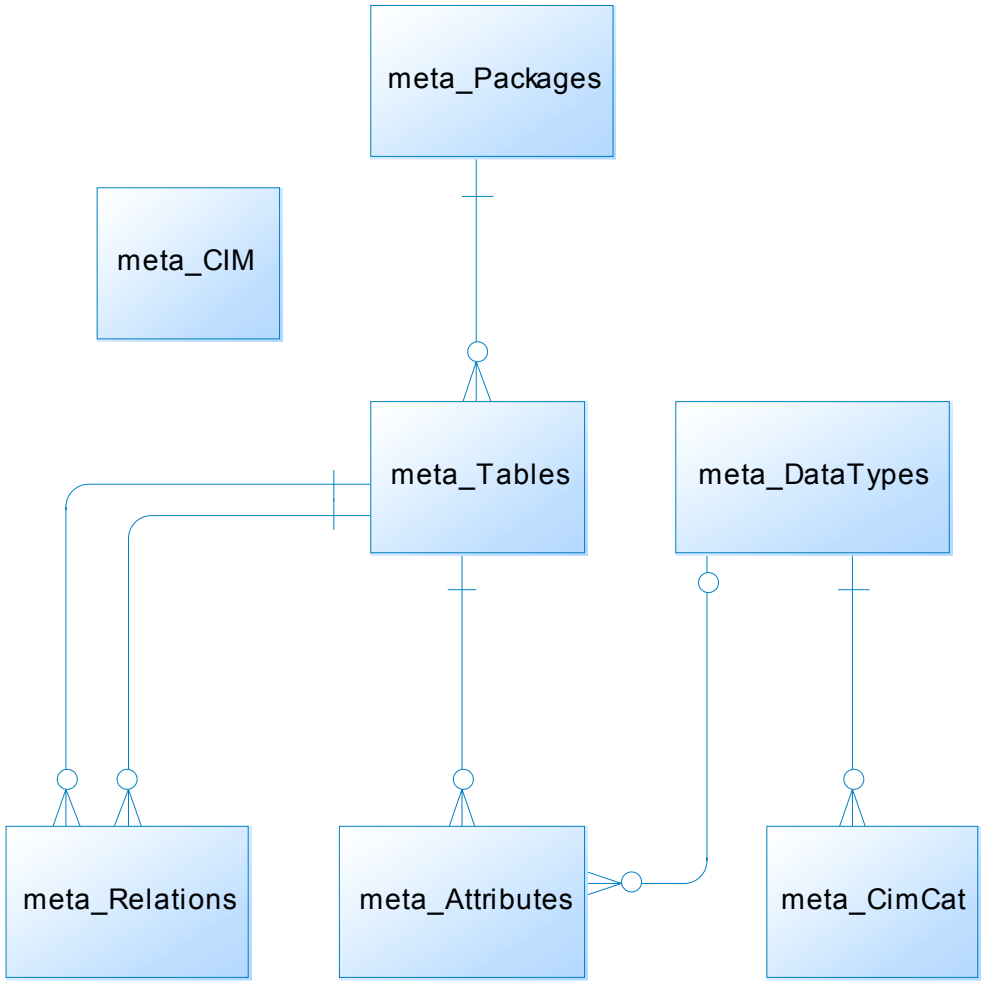
- *Više tablica povezanih relacijama 1..**

- *Više tablica povezanih relacijama 1..1*

Podaci za razmjenu

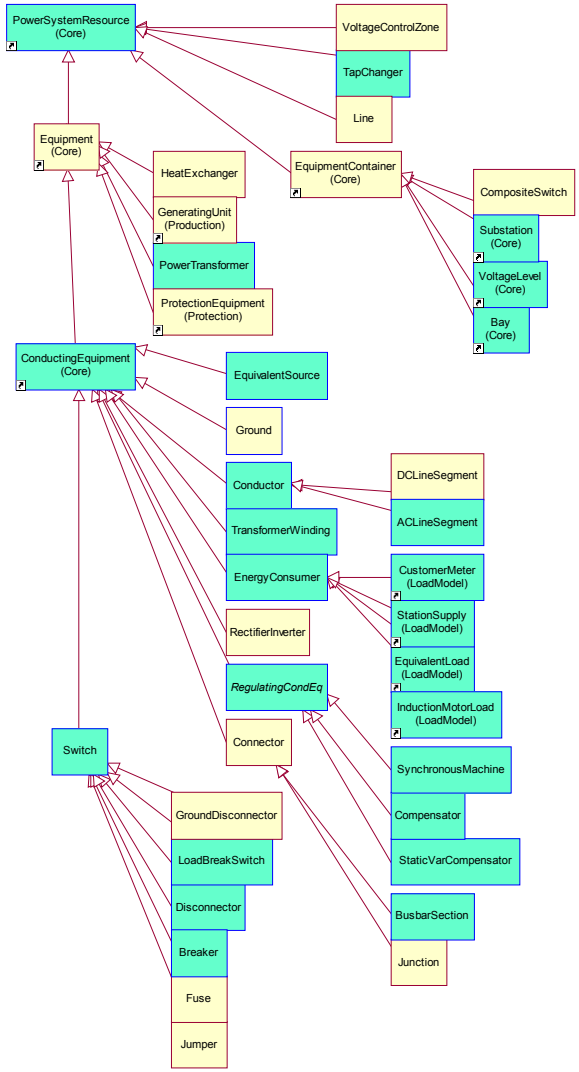
```
<cim:Substation ID="Mracl1"
cim:PowerSystemResourceName="Mracl1">
      <cim:MemberOfCompany resource="#HEP">
      <cim:Contain>
            <cim:Breaker ID="Q1"
            cim:PowerSystemResourceName="Q1"
            cim:Manufacturer="Koncar"
            cim:NormalOpen="true"/>
      </cim:Contain>
</cim:Substation>
<cim:Company ID="HEP" CompanyName="HEP OPS" >
      <cim:CompanyDescription>
            Hrvatska elektroprivreda – Operator prijenosnog sustava
      </cim:CompanyDescription>
</cim:Company>
```

CIM metamodel



Hijerarhija klasa

Class	Parentclass	Position	Package
Naming	Naming	0	Core
BasePower	BasePower	0	Core
BaseVoltage	BaseVoltage	0	Core
CurveSchedule	Naming	1	Core
PSRType	Naming	1	Core
CurveSchedFormula	Naming	1	Core
Unit	Naming	1	Core
Terminal	Naming	1	Core
Company	Naming	1	Core
PowerSystemResource	Naming	1	Core
CurveSchedData	Naming	1	Core
SubControlArea	PowerSystemResource	2	Core
ControlHouseEquipment	PowerSystemResource	2	Core
EquipmentContainer	PowerSystemResource	2	Core
Equipment	PowerSystemResource	2	Core
VoltageLevel	EquipmentContainer	3	Core
Substation	EquipmentContainer	3	Core
Bay	EquipmentContainer	3	Core
ConductingEquipment	Equipment	3	Core
SteamSupply	PowerSystemResource	2	Generation Dynamics
PrimeMover	PowerSystemResource	2	Generation Dynamics
CTTempMWCurve	CurveSchedule	2	Generation Dynamics
SteamTurbine	PrimeMover	3	Generation Dynamics



Primjer 1: Atributi entiteta

Class	Attribute	Type
BasePower	ID	PK
BasePower	basePower	ApparentPower
BaseVoltage	BasePower_ID	PK
BaseVoltage	ID	PK
BaseVoltage	nominalVoltage	Voltage
Bay	Substation_ID	PK
Bay	VoltageLevel_ID	PK
Bay	ID	PK
Bay	Cat_busBarConfiguration	BusbarConfiguration
Bay	Cat_breakerConfiguration	BreakerConfiguration
Bay	bayEnergyMeasFlag	Boolean
Bay	bayPowerMeasFlag	Boolean

```
create or replace function f_attributes(p_table VARCHAR2)
return VARCHAR2
is
v_table VARCHAR2(30);
v_pos NUMBER;
v_ind NUMBER;
v_select VARCHAR2(2000);
begin
  select position into v_pos from meta_tables
   where tablerealname = p_table;

  v_select := '''' || UPPER(p_table) || '''';

  for v_ind in 1..v_pos loop
   select parenttablename into v_table from meta_tables
    where tablename = upper(v_table);
   v_select:= v_select || ', ' || '''' || upper(v_table) || '''';
  end loop;

  v_select:= 'select distinct attributerealname from
meta_attributes '
          || ' where tablename in (' || v_select || ')'
          || ' order by attributerealname';
  return v_select;
end;
```

ATTRIBUTEREALNAME
aliasName
bayEnergyMeasFlag
bayPowerMeasFlag
Cat_breakerConfiguration
Cat_busBarConfiguration
description
ID
name
pathName
PSRType_ID
Substation_ID
VoltageLevel_ID

```
select distinct attributerealname from meta_attributes  where tablename in ('BAY',
'EQUIPMENTCONTAINER', 'POWERSYSTEMRESOURCE', 'NAMING') order by attributerealname
```

Primjer 1: Hijerarhijski upit

```
create or replace view v_classhierarchy as
select connect_by_root tablerealname leafclass, parentclass, position
  from
(select position, DECODE(parentclass, classname, NULL, classname) parentclass, classname
from v_classes)
  connect by prior parentclass = classname
order by 1, 3 desc
```

	LEAFCLASS	PARENTCLASS	POSITION
	Bay	Bay	3
	Bay	EquipmentContainer	2
	Bay	PowerSystemResource	1
	Bay	Naming	0
	Breaker	Breaker	5
	Breaker	Switch	4
	Breaker	ConductingEquipment	3
	Breaker	Equipment	2
	Breaker	PowerSystemResource	1
	Breaker	Naming	0
	BusbarSection	BusbarSection	5
	BusbarSection	Connector	4

```
create or replace view v_attributes as
select leafclass, attributerealname, classname, position  from v_classhierarchy, meta_attributes
where upper(tablerealname) = classname
order by leafclass, position desc, attributerealname
```

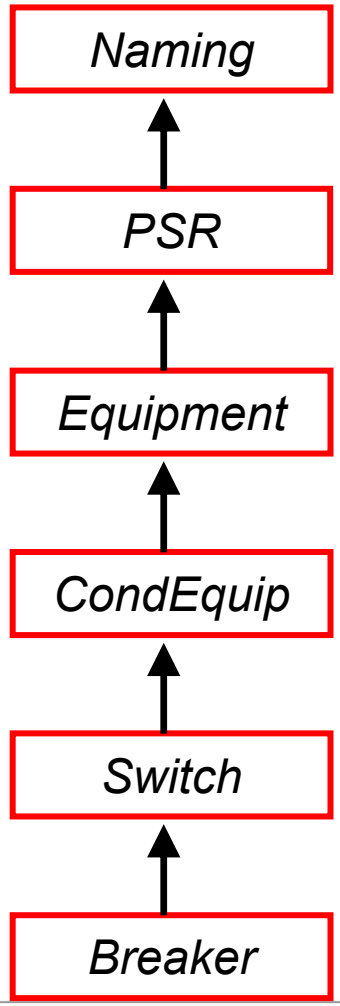
	LEAFCLASS	ATTRIBUTEREALNAME	CLASSNAME	POSITION
	Bay	bayEnergyMeasFlag	Bay	3
	Bay	bayPowerMeasFlag	Bay	3
	Bay	Cat_breakerConfiguratic	Bay	3
	Bay	Cat_busBarConfiguratior	Bay	3
	Bay	ID	Bay	3
	Bay	Substation_ID	Bay	3
	Bay	VoltageLevel_ID	Bay	3
	Bay	ID	EquipmentContair	2
	Bay	ID	PowerSystemRes	1
	Bay	PSRType_ID	PowerSystemRes	1
	Bay	aliasName	Naming	0
	Bay	description	Naming	0
	Bay	ID	Naming	0
	Bay	name	Naming	0
	Bay	pathName	Naming	0
	Breaker	ampRating	Breaker	5
	Breaker	ID	Breaker	5
	Breaker	inTransitTime	Breaker	5
	Breaker	Cat_normalOpen	Switch	4

```
select distinct attributerealname from v_classhierarchy
where leafclass = 'Bay'
order by 1
```

	ATTRIBUTEREALNAME
▶	aliasName
	bayEnergyMeasFlag
	bayPowerMeasFlag
	Cat_breakerConfiguration
	Cat_busBarConfiguration
	description
	ID
	name
	pathName
	PSRType_ID
	Substation_ID
	VoltageLevel_ID

Primjer 2: Putanja klasa

Naming.PowerSystemResource.Equipment.ConductingEquipment.Switch.Breaker




```
vl_tablename:= 'Breaker';
  vl_result:= vl_tablename;

   loop
        select parenttablename, position into vl_parenttablename,
         vl_position from meta_tables
         where upper(tablename) = upper(vl_tablename);
        vl_tablename:= vl_parenttablename;
        vl_result:= vl_tablename || '.' || vl_result;
      exit when vl_position = 1;
    end loop;
```

Primjer 2: Hijerarhijski upit

```
CREATE OR REPLACE VIEW V_CLASSHIERARCHY
(POSITION, TABLEREALNAME, LEAFCLASS)
AS
select position, tablerealname, connect_by_root tablerealname leafclass
  from (select position, tablerealname
DECODE(parenttablename, tablerealname, NULL, parenttablename)  parenttablename
 from meta_tables where instr(tablename, '_') = 0)
connect by prior parenttablename = tablerealname
```

Position	TableName	ParentTableName
5	Breaker	Switch
4	Switch	ConductingEquipment
3	ConductingEquipment	Equipment
2	Equipment	PowerSystemResource
1	PowerSystemResource	Naming
0	Naming	Naming

```
create or replace function f_get_classpath return VARCHAR2
is
 vl_result VARCHAR2(255);
begin
 for c_classes in
  (select position, tablerealname from v_classhierarchy
    where leafclass = 'Breaker‘   order by position) loop
    if c_classes.position = 0 then
      vl_result:= vl_result || c_classes.tablerealname;
    else
      vl_result:= vl_result || '.' || c_classes.tablerealname;
    end if;
  end loop;
  return vl_result;
end;
```

O čemu smo govorili

I. UVOD

II. HIJERARHIJSKI UPITI

III. EXPLAIN PLAN TABLICA

IV. COMMON INFORMATION MODEL

V. ZAKLJUČAK