

## Lösung zu Aufgabe 6

1. Ihre Aufgabe ist es nun, diese Daten in das bestehende Datenbank-Schema zu integrieren und dazu die entsprechenden Tabellen zu erstellen.

Abgabe: neue Tables

2 P

### Lösung:

```
CREATE TABLE go_term (
  id int NOT NULL,
  name varchar(255) NOT NULL default '',
  term_type varchar(55) NOT NULL default '',
  acc varchar(32) NOT NULL default '',
  is_obsolete int NOT NULL default '0',
  is_root int NOT NULL default '0',
  PRIMARY KEY (id)
);

CREATE TABLE go_term2term (
  id int NOT NULL,
  relationship_type_id int NOT NULL default '0',
  term1_id int NOT NULL default '0',
  term2_id int NOT NULL default '0',
  PRIMARY KEY (id)
);

--CREATE TABLE go_graph_path (
--  term1_id int NOT NULL default '0',
--  term2_id int NOT NULL default '0',
--  distance int NOT NULL default '0',
--);

CREATE TABLE go_temp (
go_identifier VARCHAR );

CREATE TYPE go_term_ancestor AS (ancestor_id integer,
                                distance integer);

COPY go_term FROM '/tmp/term.txt';

COPY go_term2term FROM '/tmp/parent_child.txt';

COPY go_temp FROM '/tmp/GO-Acc.txt';
```

2. Wie auch bei QuickGO und anderen Retrieval Systems sollen wir in der Lage sein, mit einer Anfrage durch den Nutzer alle ancestors (Vorgänger) eines bestimmten GO-terms abrufen zu können. Ermitteln Sie die Vorgänger mit Hilfe einer Datenbank-Funktion(PL/pgSQL). Innerhalb der Vorgänger eines GO-terms muß eine Ordnung herrschen.

Abgabe: Funktion zur Ermittlung der Vorgänger

**4 P**

```
CREATE or replace FUNCTION get_go_parentlist (int, int) RETURNS setof
go_term_ancestor AS
'
DECLARE
    go_start ALIAS FOR $1;
    go_dist ALIAS FOR $2;
    res RECORD;
    help RECORD;
BEGIN
    FOR res IN SELECT term1_id, go_dist+1 AS distance FROM go_term2term WHERE
term2_id = go_start LOOP
        FOR help IN SELECT ancestor_id, distance FROM
get_go_parentlist(res.term1_id, go_dist+1) LOOP
            RETURN NEXT help;
        END LOOP;
        RETURN NEXT res;
    END LOOP;
    RETURN;
END
' LANGUAGE 'plpgsql';
```

```
CREATE OR REPLACE FUNCTION fill_go_graph_path() RETURNS text AS'
DECLARE
    childs RECORD;
    help RECORD;
BEGIN
    FOR childs IN SELECT id FROM go_term LOOP
        INSERT INTO go_graph_path SELECT childs.id, ancestor_id, distance from
get_go_parentlist(childs.id, 0);
    END LOOP;
    RETURN ''fertig'';
END
' LANGUAGE 'plpgsql';
```

```
SELECT fill_go_graph_path();
```

3. Für das Protein mit dem Swiss-Prot Identifier 'DHA4\_HUMAN' sind in den Datenbank-Referenzen GO-Accession Numbers gespeichert. Zu diesen wollen Sie die Namen, und für jede GO-Accession-numbers eine Liste von deren Vorgängern. Es sollen neben der GO-Accession Numbers, des Namens und des Term typen der eigentlichen GO-Accession Numbers auch noch die Namen der Vorgänger und deren Abstand zu der gegebenen GO-Accession ausgegeben werden (order by term\_type, dist!).

Abgabe: Query und Ergebnis

4 P

```
SELECT t1.acc, t1.name, t1.term_type, t2.id, t2.name, gp.distance
FROM swissprot sp, sp_reference sr, go_term t1, go_term t2, go_graph_path gp
WHERE sp.identiflier='DHA4_HUMAN'
      AND sp.id = sr.swiss_id
      AND sr.databank = 'GO'
      AND sr.accession = t1.acc
      AND t1.id = gp.term1_id
      AND gp.term2_id = t2.id
ORDER BY t1.term_type, gp.distance DESC, t2.id;
```

acc	name	term_type	id	name	distance
GO:0008544	epidermal differentiation	biological_process	1	Gene_Ontology	7
GO:0007417	central nervous system development	biological_process	1	Gene_Ontology	6
GO:0007422	peripheral nervous system development	biological_process	1	Gene_Ontology	6
GO:0008544	epidermal differentiation	biological_process	5390	biological_process	6
GO:0007417	central nervous system development	biological_process	5390	biological_process	5
GO:0007422	peripheral nervous system development	biological_process	5390	biological_process	5
GO:0008544	epidermal differentiation	biological_process	7607	development	5
GO:0006629	lipid metabolism	biological_process	1	Gene_Ontology	4
GO:0007417	central nervous system development	biological_process	7607	development	4
GO:0007422	peripheral nervous system development	biological_process	7607	development	4
GO:0008544	epidermal differentiation	biological_process	7851	morphogenesis	4
GO:0006629	lipid metabolism	biological_process	5390	biological_process	3
GO:0007417	central nervous system development	biological_process	7851	morphogenesis	3
GO:0007422	peripheral nervous system development	biological_process	7851	morphogenesis	3
GO:0008544	epidermal differentiation	biological_process	7858	organogenesis	3
GO:0007417	central nervous system development	biological_process	7858	organogenesis	2
GO:0007422	peripheral nervous system development	biological_process	7858	organogenesis	2
GO:0008544	epidermal differentiation	biological_process	7900	histogenesis	2
GO:0006629	lipid metabolism	biological_process	8464	physiological processes	2
GO:0008544	epidermal differentiation	biological_process	7901	ectoderm development	1
GO:0007417	central nervous system development	biological_process	7968	neurogenesis	1
GO:0007422	peripheral nervous system development	biological_process	7968	neurogenesis	1
GO:0006629	lipid metabolism	biological_process	8537	metabolism	1

(23 rows)

4. Wir bekommen eine Liste von GO-Accession Numbers (GO\_Acc.txt). Speichern sie diese Liste als Tabelle in der Datenbank, um darauf von einer Datenbank-Funktion zugreifen zu können. Die GO-Accession number GO:0016296 hat mit den GO-Accession Numbers von GO\_Acc.txt gemeinsame Knoten. Berechnen Sie den/die kleinsten gemeinsamen Knoten zwischen GO:0016296 und jeder GO-Accession number in GO\_Acc.txt mit Hilfe einer Datenbank-Funktion. Geben Sie neben der kürzesten Distanz auch die GO-Accession number und den Namen des gemeinsamen Knotens an.

Abgabe: Funktion und deren Ergebnis

5 P

**Lösung:**

```
select distinct mn.go_identifizier as start_point, t.acc as common_knot, t.name,
t.term_type, mn.distance
from go_term t
JOIN (SELECT tmp.go_identifizier, gp.term2_id, gp.distance as dist1
      FROM go_temp tmp, go_term gt1, go_graph_path gp
      WHERE tmp.go_identifizier = gt1.acc
           AND gt1.id = gp.term1_id ) as list1
on t.id = list1.term2_id
JOIN (select ggp.term2_id, ggp.distance AS dist2
      from go_term gt2, go_graph_path ggp
      where gt2.acc = 'GO:0016296'
           and gt2.id = ggp.term1_id) AS knots1
ON knots1.term2_id = list1.term2_id
JOIN (select list.go_identifizier, list.term1_id, min(list.dist1 + knots.dist2) as
distance
      from (SELECT tmp.go_identifizier, gp.term1_id, gp.term2_id, gp.distance as
dist1
           FROM go_temp tmp, go_term gt1, go_graph_path gp
           WHERE tmp.go_identifizier = gt1.acc
                AND gt1.id = gp.term1_id ) as list
      JOIN (select ggp.term2_id, ggp.distance AS dist2
           from go_term gt2, go_graph_path ggp
           where gt2.acc = 'GO:0016296'
                and gt2.id = ggp.term1_id) AS knots
           ON knots.term2_id = list.term2_id
      group by list.go_identifizier, list.term1_id) as mn
ON (mn.go_identifizier = list1.go_identifizier
    and mn.distance = (knots1.dist2 + list1.dist1))
```

start_point	common_knot	name	term_type	distance
GO:0000703	GO:0016788	hydrolase activity, acting on ester bonds	molecular_function	7
GO:0003812	GO:0016787	hydrolase activity	molecular_function	8
GO:0003829	GO:0003824	enzyme activity	molecular_function	10
GO:0003829	GO:0016740	transferase activity	molecular_function	10
GO:0003849	GO:0003824	enzyme activity	molecular_function	9
GO:0003860	GO:0016790	thiolester hydrolase activity	molecular_function	4
GO:0004406	GO:0008415	acyltransferase activity	molecular_function	6
GO:0004483	GO:0003824	enzyme activity	molecular_function	10
GO:0004483	GO:0016740	transferase activity	molecular_function	10
GO:0004485	GO:0003824	enzyme activity	molecular_function	9
GO:0004498	GO:0003824	enzyme activity	molecular_function	8
GO:0004590	GO:0003824	enzyme activity	molecular_function	9
GO:0004671	GO:0003824	enzyme activity	molecular_function	10
GO:0004671	GO:0016740	transferase activity	molecular_function	10
GO:0008877	GO:0016788	hydrolase activity, acting on ester bonds	molecular_function	7
GO:0009030	GO:0003824	enzyme activity	molecular_function	7
GO:0015610	GO:0003674	molecular_function	molecular_function	10
GO:0016912	GO:0003674	molecular_function	molecular_function	11
GO:0016912	GO:0003824	enzyme activity	molecular_function	11
GO:0046429	GO:0003824	enzyme activity	molecular_function	6

(20 rows)

5. Von dieser Liste (GO-Acc.txt) sollen alle gemeinsame Knoten aller mit Hilfe einer Datenbank-Funktion errechnet werden und diese mit GO-Accession, dem Namen und dem Term Type ausgegeben werden. Es soll außerdem erkennbar sein, welcher der kleinste gemeinsame Knoten ist.

Abgabe: Function und das Ergebnis

5 P

### Lösung mit SQL:

```
SELECT t.acc, t.name, t.term_type, max(g.distance) AS depth
FROM go_term t, go_graph_path g, go_term tmp, go_term gt
WHERE tmp.go_identifier = gt.acc
      AND gt.id = g.term1_id
      AND t.id = g.term2_id
GROUP BY t.acc, t.name, t.term_type
HAVING count(g.term1_id) >= (SELECT count(*) FROM go_term)
ORDER BY depth ASC, acc;
```

acc	name	term_type	depth
GO:0003824	enzyme activity	molecular_function	10
GO:0003674	molecular_function	molecular_function	12
GO:0003673	Gene_Ontology	Gene_Ontology	13

(3 rows)

Die Berechnungen der Vorgänger soll möglichst schnell sein (Tip: Beziehungen vorberechnen und speichern)