Stefan Selent & Christopher Schiefer [Gruppe1] 22.05.2016

Informationsintegration SS16: Prof. Ulf Leser & Yvonne Lichtblau

Übung 2: Database Setup / Basic Queries

Create Table Statements

```
create table gene_info(
      tax_id integer,
      gene_id integer,
      symbol varchar(80),
      chromosome varchar(200),
      map_location varchar(50),
      status varchar(11),
      protein_accession_version varchar(15),
      protein_gi integer,
      start_pos integer,
      end_pos integer
);

create table gene2go(
      tax_id integer,
      gene_id integer,
      go_id varchar(10),
      evidence varchar(5)
);

create table synonym(
      gene_id integer,
      name varchar(200)
);
```

Queries & Answers

Query 1: How many genes does your gene table have?

gruppe1=> SELECT COUNT(DISTINCT gene_id) FROM gene_info;

count

13886307

Query 2: How many relationships between genes and a GO term are there?

gruppe1=> SELECT COUNT(*) FROM (SELECT DISTINCT gene_id, go_id FROM gene2go) AS rel;

count

1621602

Query 3: How many distinct GO terms are annotated to at least one gene?

gruppe1=> SELECT COUNT(DISTINCT go_id) FROM gene2go WHERE gene_id IS NOT NULL;

count

24774

Query 4: How many gene names are present (symbol or snyonym)?

gruppe1=> SELECT COUNT(DISTINCT identifier) FROM (SELECT DISTINCT symbol FROM gene_info UNION SELECT DISTINCT name FROM synonym2) AS identifier;

count

13206991

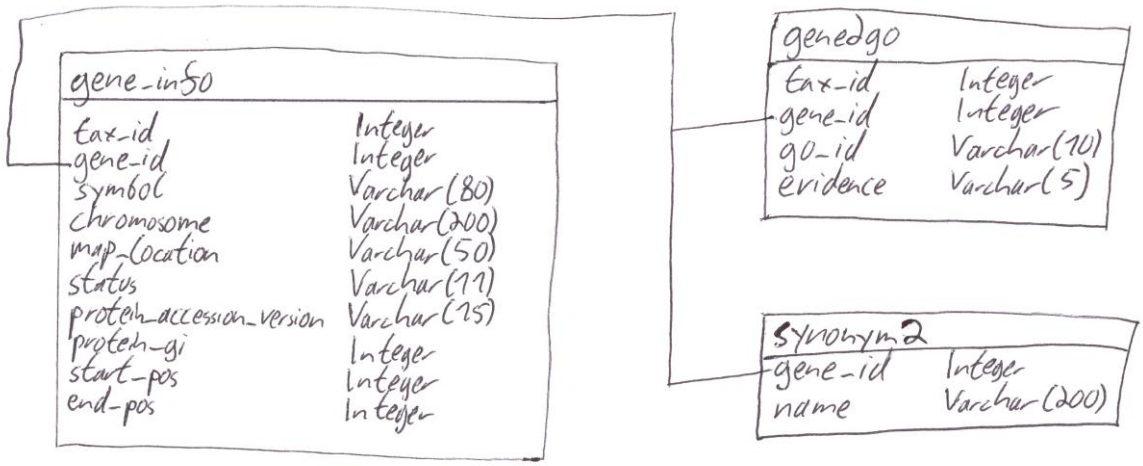
Query 5: How many synonyms are assigned to more than one gene?

gruppe1=> SELECT COUNT(*) FROM (SELECT name, COUNT(*) AS counter FROM synonym2 GROUP BY name) AS recs WHERE counter > 1;

count

72701

SQL Table Schema



Workflow Description

Zur Vorbereitung des Importierens in die Datenbank haben wir die Dateien zuerst mit Python bzw. Bash gekürzt und entsprechend zusammengefügt. Damit waren die Daten dann beim Einfügen in SQL bereits im richtigen Format und in der Datenbank selber war keine Änderung mehr nötig.

gene2go

Anpassung des Headers:

```
sed -i '1s/.*/tax_id  gene_id    go_id evidence   Qualifier  GO_term
      PubMed     Category/' gene2go
```

Entfernen irrelevanter Spalten:

```
cut -d '    ' -f5-8 --complement gene2go > gene2go_filtered
```

Importieren in SQL:

```
\COPY gene2go FROM './gene2go_filtered' WITH NULL '-' DELIMITER '
           ' CSV HEADER;
```

synonym2

Herausschreiben der Synonyme aus der *gene_info* Tabelle mit Python, mit je einer resultierenden Zeile pro Synonym:

```
synTuples = []
with open("gene_info", "r") as infoFile:
    infoFile.next()
    infoFile.next()
    for row in infoFile:
        rSplit = row.split("\t")
        if rSplit[4] != "-" and rSplit[1] != "-":
            syns = rSplit[4].split("|")
            for s in syns:
                row = rSplit[1]+"\t"+s
                synTuples.append(row)

outFile = open("syns.csv", "w")
outFile.write("\n".join(synTuples))
outFile.close
```

Und anschließendem Import in SQL:

```
\copy synonym2 FROM '.syns.csv' WITH NULL '-' DELIMITER E'\t' CSV;
```

gene_info

Mit Python wurden erst die relevanten Spalten aus der gegebenen Tabelle gene_info in einem Dictionary gespeichert, welches die GeneID zusammen mit der Tax_ID als Schlüssel nimmt. Anschließend wird die Tabelle gene2refseq durchgegangen und die Zeilen mit den Informationen aus gene_info ergänzt, wenn zu der entsprechenden GeneID ein Eintrag im Dictionary gespeichert ist. Ansonsten werden die Werte mit NULL ergänzt und ausgegeben. Schlussendlich werden die Tupel aus gene_info, die nicht mit refseq-Daten erweitert wurden, mit NULL-Werten in den entsprechenden Spalten ausgegeben.

```
## actual content, organize entries in dictionary
table_cont = {}
infoGenes = set()

## 14,023,138 lines
## -- remove first two lines
with open("gene_info", "r") as infoFile:
    ## columns:
    ## 0: tax_id
    ## 1: GeneID
    ## 2: symbol
    ## 6: chromosome
    ## 7: map_location
```

```
    next(infoFile)
    next(infoFile)
    for row in infoFile:
        rSplit = row.split("\t")
        dbTup = []
        dbTup.append(rSplit[2]) ## symbol
        dbTup.append(rSplit[6]) ## chromosome
        dbTup.append(rSplit[7]) ## map_location
        table_cont[(rSplit[0], rSplit[1])] = dbTup
        infoGenes.add((rSplit[0], rSplit[1]))

## 22,162,843 lines
with open("gene2refseq", "r") as infoFile:

    with open("gene_comb_table.csv", "w") as out:
        ## columns:
        ## 0: tax_id
        ## 1: GeneID
        ## 2: status
        ## 5: protein_accession_version
        ## 6: protein_gi
        ## 9: start_pos
        ## 10: end_pos
        next(infoFile)
        cnt = 0
        first = True
        for row in infoFile:
            dbTup = []
            rSplit = row.split("\t")
            if (rSplit[0], rSplit[1]) in table_cont:
                ## gene was already in first table
                dbTup = [elmnt for elmnt in table_cont[(rSplit[0],
rSplit[1])]]
                dbTup.append(rSplit[2]) # status
                dbTup.append(rSplit[5]) # protein_accession_version
                dbTup.append(rSplit[6]) # protein_gi
                dbTup.append(rSplit[9]) # start_pos
                dbTup.append(rSplit[10]) # end_pos
                if (rSplit[0], rSplit[1]) in infoGenes:
                    infoGenes.remove((rSplit[0], rSplit[1]))
            else:
                ## gene was not in first table
                ## fill first three items with NULL
                dbTup.append("-")
                dbTup.append("-")
                dbTup.append("-")
                dbTup.append(rSplit[2]) # status
                dbTup.append(rSplit[5]) # protein_accession_version
                dbTup.append(rSplit[6]) # protein_gi
                dbTup.append(rSplit[9]) # start_pos
                dbTup.append(rSplit[10]) # end_pos

            dbTup.insert(0, rSplit[1]) ## insert geneID
            dbTup.insert(0, rSplit[0]) ## insert tax_ID
            out.write("\t".join(dbTup))
            out.write("\n")

## write remaining genes that were only in first table
with open("gene_comb_table.csv", "a") as out:
    for k in infoGenes:
        tup = table_cont[k]
        if len(tup) < 8:
            print "yes, needed"
```

```
            ## gene was only in first table, not in second
            tup.append("-")
            tup.append("-")
            tup.append("-")
            tup.append("-")
            tup.append("-")
        tup.insert(0, k[1]) ## insert geneID
        tup.insert(0, k[0]) ## insert tax_ID
        out.write("\t".join(tup))
        out.write("\n")
```

Und anschließend in SQL importiert:

```
\copy gene_info FROM '.gene_comb_table.csv' WITH NULL '-' DELIMITER
E'\t' CSV;
```