

# Molekularbiologische Datenbanken

Übungen  
Einleitung 4

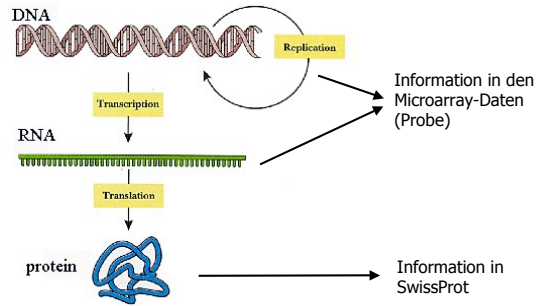


Silke Trißl  
Prof. Ulf Leser



Wissensmanagement in der  
Bioinformatik

## Aufgabe 4 – Integrieren von SwissProt-Daten



Silke Trißl, Prof. Ulf Leser Molekularbiologische Datenbanken, Übung, SoSe 2004

2

## Informationen aus SwissProt

Feld	Beschreibung	Kardinalität
ID	SwissProt Identifikation	1
AC	Accession Nummer	n
DE	Description	1
DT	Datum	1 created 1 annotation update 1 sequence update
DR	Database reference	n
FT	Features	n
SQ	Sequence	1

<http://www.expasy.org/sprot/userman.html>



Silke Trißl, Prof. Ulf Leser Molekularbiologische Datenbanken, Übung, SoSe 2004

3

## SwissProt Eintrag

```

ID HBB HUMAN STANDARD; PRT; 146 AA.
AC P02023; Q13852; Q14481; Q14510; Q9BX96; Q9UCP8; Q9UCP9;
DE Hemoglobin beta chain.
GN HBB.
OS Homo sapiens (Human),
   Pan troglodytes (Chimpanzee), and
   Pan paniscus (Pygmy chimpanzee) (Bonobo).
OC Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
   Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
OX NCBI_TaxID=9606, 9598, 9597;
RN [1]
RP SEQUENCE.
RC SPECIES=Human;
RA Braunlitzer G., Gehring-Muller R., Hilschmann N., Hille K., Hobom G.,
   Rudloff V., Wittmann-Liebold B.;
RT "The constitution of normal adult human haemoglobin.";
RL Hoppe-Seyler's Z. Physiol. Chem. 325:283-286(1961).
RH [2]
RP SEQUENCE FROM N.A.
RC SPECIES=Human;
RX MEDLINE=8166467; PubMed=6254664;
RA Lawn R.M., Efstratiadis A., O'Connell C., Maniatis T.;
RT "The nucleotide sequence of the human beta-globin gene.";
RL Cell 21:647-651(1980).
RN [3]
RP .....
    
```

Liste von AC, aus denen der Eintrag hervorgegangen ist



Silke Trißl, Prof. Ulf Leser Molekularbiologische Datenbanken, Übung, SoSe 2004

4

## SwissProt Eintrag

```

...
RL Hemoglobin 14:555-557(1990).
CC -!- FUNCTION: Involved in oxygen transport from the lung to the
   various peripheral tissues.
...
DR EMBL; D01317; AAA16334.1; -.
...
DR PIR; D93303; HBCZP.
DR PDB; 1A00; 18-MAR-98.
...
DR GO; GO:0005344; P:oxygen transporter activity; TAS.
DR GO; GO:0015671; P:oxygen transport; NAS.
DR InterPro; IPR002337; Beta haem.
...
KW Heme; Oxygen transport; Transport; Erythrocyte; Acetylation;
   S-nitrosylation; Disease mutation; Polymorphism; 3D-structure.
...
FT INTP_MEST 0 0 Iron (heme distal ligand).
FT METAL 63 63 Iron (heme proximal ligand).
FT MOD_RES 1 1 ACETYLATION (IN VARIANT RALEIGH).
FT CARBOHYD 1 1 N-linked (Glc) (glycation); in Hb A1c.
FT MOD_RES 93 93 S-NITROSYLATION.
FT VARIANT 1 1 V -> A (in Raleigh; O2 affinity down).
FT /FTID=VAR_002856.
...
FT RELIX 124 142
FT TURN 143 144
SQ SEQUENCE 146 AA; 15867 MW; EACBC70CFD466A1 CRC64;
   VILTFPEKSA VTLALGKRVNV DEVGGEALGR LLVYVPTQR FFEFSDLST PDAVGNPKV
   KARKKRVLEA FSGGLANLW LKQTATLSE LHCCKLVNDF ENPRLGNVL VCVLAKHFK
   EFTFPVQAAV QKVYAGVANA LAKHYH
    
```

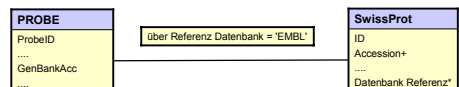
Features mit Art des Features, Startposition, Endposition, Kommentar



Ende eines Eintrags

5

## Verbindung Array - SwissProt



Silke Trißl, Prof. Ulf Leser Molekularbiologische Datenbanken, Übung, SoSe 2004

6

## Aufgabe 4

- Integrieren von SwissProt
  - Erstellen des konzeptuellen Modells
  - Erstellen des relationalen Modells
- unter Berücksichtigung von Versionen
  - Versionierung von SwissProt
- SQL Anfragen auf der erweiterten Datenbank stellen



## Versionierung

- Das Wissen wächst ständig
  - Ensembl: ca. 50% Änderungen pro Release
  - UniGene: Gene verschieben sich mit neuen ESTs
- Analysen benutzen bestimmte Version (aktuellste)
- **Nachvollziehbarkeit von Analysen nicht gewährleistet**
- **Versionierung von Daten essentiell**
  - Version der Datenbank (Releases)
  - Versionen von Objekten / Attributen
- **Verschiedene Aspekte**
  - Speichern von verschiedenen Versionen
  - Zugriff auf (alte) Versionen
  - Eingrenzen von Änderungen
  - Referenzieren auf Versionen



## Versionierung

- Einträge können von Version 1 zu Version 2
  - unverändert,
  - geändert (update),
    - -> Identifizierung von Änderungen
  - hinzugefügt,
  - gelöscht
    - 2 Einträge werden zu einem zusammengefasst -> mindestens ein Eintrag muss gelöscht werden



## Versionierung

```
Eintrag am 04.04.2003                               Eintrag am 29.11.2002
ID KIF2C_HUMAN STANDARD; PRT; 725 AA_ID MCAK_HUMAN STANDARD; PRT; 725
AC Q99661; AC Q99661;
DT 28-FEB-2003 (Rel. 40, Created) DT 15-JUN-2002 (Rel. 40, Created)
DT 28-FEB-2003 (Rel. 41, Last sequence update) DT 15-JUN-2002 (Rel. 41, Last sequence update)
DT 15-SEP-2003 (Rel. 42, Last annotation update) DT 15-JUN-2002 (Rel. 41, Last annotation update)
DE Kinesin-like protein KIF2C (Mitotic centromere M Mitotic centromere-associated kinesin (M
DE (MCAK) (Kinesin-like protein 6). GN KNSL6. OS Homo sapiens (Human).
OS Homo sapiens (Human). OS Homo sapiens (Human).
OC Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Mammalia; Eutheria; Primates; Catarrhini; HOCOC
OC Mammalia; Eutheria; Primates; Catarrhini; HOCOC Mammalia; Eutheria; Primates; Catarrhini; HOCOC
OX NCBI_TaxID=9606; OX NCBI_TaxID=9606;
RN [1] RN [1]
RF SEQUENCE FROM N.A. (ISOFORM 1). RF SEQUENCE FROM N.A.
RC TISSUE=T-cell; RC TISSUE=T-CELL;
RX MEDLINE=98094213; PubMed=9434124; RX MEDLINE=98094213; PubMed=9434124;
RA Kim I.-G., Jun D.Y., Sohn U., Kim Y.H.; RA Kim I.-G., Jun D.Y., Sohn U., Kim Y.H.;
RT "Cloning and expression of human mitotic cenRT "Cloning and expression of human mitotic
RT gene."; RT gene.";
RL Biochim. Biophys. Acta 1359:181-186(1997). RL Biochim. Biophys. Acta 1359:181-186(1997)
....
```



## Versionen im relationalen Modell

- Lineare, tuple-basiert, mit kompletten Versionen
- Anforderungen: Zugriff auf
  - aktuelle Version
  - Zustand der Datenbank zu beliebigem Zeitpunkt  $t_0$
- 2 Varianten
  - Single Table
  - Schattentabellen
- Annahme: **Keine Schlüsselwiederverwertung**
  - Schlüssel K wird niemals für verschiedene Objekten benutzt
  - Kann z.B. durch Sequenz sichergestellt werden



## Variante 1: Single-table

- Erweiterung jeder Tabelle T um Attribute
  - Versionsnummer V
  - ALIVE Flag A
  - VALIDFROM D
  - Schlüsselveränderung  $K \rightarrow (K+V)$



## Variante 1: INSERT

- **INSERT** Objekt K in T
  - Gibt es K schon in T ?
  - Nein
    - INSERT K INTO T (V=0, A=T, D=SYSDATE)
  - Ja
    - Letzte Version von K in T finden ( $V_x$ )
    - INSERT K INTO T (V= $V_x+1$ , A=T, D=SYSDATE)



## Variante 1: DELETE

- **DELETE** Objekt K aus T
  - Gibt es K schon in T ?
  - Nein
    - Nichts tun
  - Ja
    - Letzte Version von K in T finden ( $K_{alt}$  mit  $V_x$ )
    - $K_{alt}.A=T$  ?
      - Ja
        - INSERT K INTO T (V= $V_x+1$ , A=F, D=SYSDATE)
      - Nein
        - Nichts tun
- Werte von K beim INSERT sind beliebig; es zählt V, D und A



## Variante 1: UPDATE

- **UPDATE** Objekt K in T
  - K in T vorhanden ?
  - Nein
    - Nichts tun
  - Ja
    - Letzte Version von K in T finden ( $K_{alt}$  mit  $V_x$ )
    - $K_{alt}.A=T$  ?
      - Ja
        - INSERT K INTO T (V= $V_x+1$ , A=T, D=SYSDATE)
      - Nein
        - Nichts tun



## Variante 1: SELECT

- **SELECT** aktuellste Version von  $k_0$
- **SELECT** alle Tupel zum Zeitpunkt  $d_0$

```
SELECT *
FROM t
WHERE a=T AND
       k=k0 AND
       d = (SELECT MAX(d)
            FROM t2
            WHERE t2.K=k0)
```

```
SELECT *
FROM t1
WHERE a=T AND
       d<=d0 AND
       d = (SELECT MAX(d)
            FROM t2
            WHERE t2.K=t1.K AND
                  t2.d<=d0)
```

Was könnte man sparen ?



## Variante 1: Bewertung

- Varianten
  - Versionsnummer weglassen (Datum reicht)
  - Markierung der aktuellsten Version hilfreich
    - INSERT erfordert 1INSERT + 1UPDATE
    - Zugriff auf aktuellste Version schneller
- Bewertung
  - INSERT / DELETE / UPDATE erfordern Trigger
  - Verlangsamung bei SELECTs (ohne ACTUAL Flag)
  - Verlangsamung durch wachsende Tabelle
  - Syntaxkomplexität durch Views abfangen



## Variante 2: Schattentabellen

- Pro Tabelle T anlegen einer Tabelle  $T^S$ 
  - Zusätzliche Attribute
    - Versionsnummer V
    - UNTIL D
  - Schlüsselveränderung  $K \rightarrow (K+V)$
- T bleibt unverändert
- $T^S$  speichert alte Versionen
- T speichert nur aktuellste Version



## Variante 2: INSERT

- **INSERT** Objekt K in T
  - K in T vorhanden ?
  - Nein
    - INSERT K in T
  - Ja: Sei dies das Tupel  $K_{alt}$ 
    - Letzte Version von K in  $T^S$  finden ( $V_x$ )
    - $V_x$  existiert: INSERT  $K_{alt}$  INTO  $T^S$  ( $V=V_x+1$ ,  $D=SYSDATE$ )
    - $V_x$  existiert nicht: INSERT  $K_{alt}$  INTO  $T^S$  ( $V=0$ ,  $D=SYSDATE$ )
    - DELETE  $K_{alt}$  FROM T
    - INSERT K INTO T
- Tupel wird von T nach  $T^S$  verschoben



## Variante 2: DELETE

- **DELETE** Objekt K aus T
  - K in T vorhanden ?
  - Nein
    - Nichts tun
  - Ja: Sei dies das Tupel  $K_{alt}$ 
    - Letzte Version von K in  $T^S$  finden ( $V_x$ )
    - $V_x$  existiert: INSERT  $K_{alt}$  INTO  $T^S$  ( $V=V_x+1$ ,  $D=SYSDATE$ )
    - $V_x$  existiert nicht: INSERT  $K_{alt}$  INTO  $T^S$  ( $V=0$ ,  $D=SYSDATE$ )
    - DELETE  $K_{alt}$  FROM T
- Objekt in T vorhanden
  - Objekt ist gültig
- Objekt nicht in T vorhanden, aber in  $T^S$ 
  - Objekt war gültig bis D



## Variante 2: UPDATE

- **UPDATE** Objekt K in T
  - K in T vorhanden ?
  - Nein
    - Nichts tun
  - Ja: Sei dies das Tupel  $K_{alt}$ 
    - Letzte Version von K in  $T^S$  finden ( $V_x$ )
    - $V_x$  existiert: INSERT  $K_{alt}$  INTO  $T^S$  ( $V=V_x+1$ ,  $D=SYSDATE$ )
    - $V_x$  existiert nicht: INSERT  $K_{alt}$  INTO  $T^S$  ( $V=0$ ,  $D=SYSDATE$ )
    - DELETE  $K_{alt}$  FROM T
    - INSERT K INTO T



## Variante 2: SELECT

- **SELECT** aktuellste Version von  $k_0$ 

```
SELECT *
FROM t
WHERE k=k0
```
- **SELECT** alle Tupel zum Zeitpunkt  $d_0$ 
  - Kompliziert ....
  - Erfordert Zugriff auf T und  $T^S$
  - Vorsicht: Datum in  $T^S$  gibt **Ende der Gültigkeit** an



## Variante 2: SELECT auf Zeitpunkt

- Das letzte Mal vor  $d_0$  geändert
    - In T (nehmen)
    - In  $T^S$ , aber kein Eintrag mit  $d > d_0$
  - Nach  $d_0$  noch geändert
    - In T (nicht nehmen)
    - In  $T^S$ , kleinsten Eintrag nehmen mit  $d > d_0$
  - Vor  $d_0$  gelöscht
    - Nicht in T
    - In  $T^S$ , aber kein Eintrag mit  $d > d_0$
- ```
SELECT *
FROM t
WHERE NOT EXISTS
  (SELECT *
   FROM t_s
   WHERE t.k=t_s.k AND
         t_s.d > d0)

UNION

SELECT *
FROM t_s
WHERE t_s.d > d0
AND EXISTS
  (SELECT *
   FROM t
   WHERE t.k=t_s.k)

AND t_s.d =
  (SELECT MIN(d)
   FROM t_s t2
   WHERE t_s.k=t2.k)

UNION
...
```



## Variante 2: Bewertung

- Bewertung
  - INSERT / DELETE / UPDATE erfordern Trigger
  - Sehr schneller Zugriff auf aktuellste Version (kein Unterschied zu nicht-versioniert)
  - Komplexer Zugriff auf Zustand zu Zeitpunkt d
  - Komplizierteres INSERT /DELETE / UPDATE
  - Gut geeignet zur Archivierung (T bleibt unangetastet)

Keine Objekte (Schlüssel) wiederbeleben !



## Vergleich

- Häufige Änderungen, eher wenig Lesezugriffe - Variante 1
- Seltenerer Änderungen, vor allem Zugriff auf aktuellste Version – Variante 2
- Variante 2 eher für MDB geeignet
- Vorsicht vor **unvorhergesehenen Effekten** (Indexdegradierung, Tabellenfragmentierung, etc.)



## Identifikation von Veränderungen

- Änderungen in den Daten erkennen
  - Speichern nur von Teilen einer Quelle
    - Änderungen von uninteressanten Attributen nicht von Bedeutung
  - Zeitbasiert
    - Die Daten haben einen 'Zeitstempel'
    - möglich für vollständige Übernahme
  - Tupelbasiert
    - Erkennen von Veränderungen innerhalb der Tuples
    - zu bevorzugen bei Teilübernahmen



## Fragen?

- Aufgabe 4 über Goya oder auf Web-page
- Folien auf der Web-page
- Lösung bis 08.06. , 17 Uhr per e-mail oder in RUD25 IV.104

