

Molekularbiologische Datenbanken

Übungen

Aufgabe 4

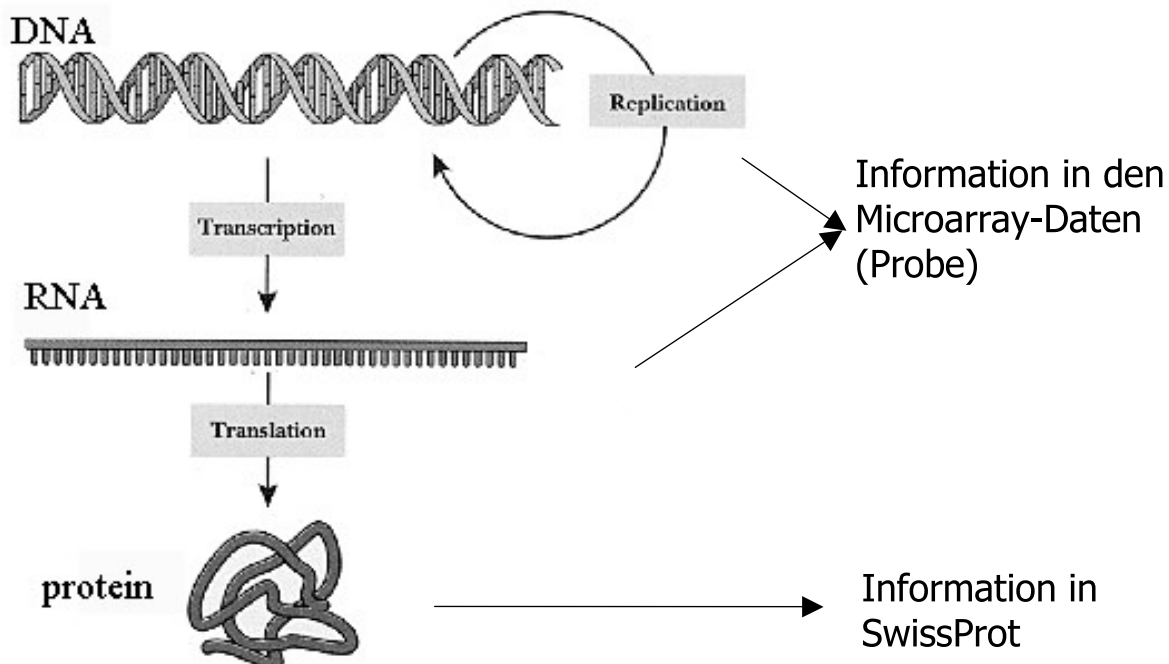


Silke Trißl
Ulf Leser



Wissensmanagement in der
Bioinformatik

Aufgabe 4 – Integrieren von SwissProt-Daten



SwissProt Eintrag

```
ID   KF2C_HUMAN          STANDARD;          PRT;    725 AA.
AC   Q99661;
DT   16-OCT-2001 (Rel. 40, Created)
DT   29-FEB-2003 (Rel. 41, Last sequence update)
DT   15-SEP-2003 (Rel. 42, Last annotation update)
DE   Kinesin-like protein KIF2C (Mitotic centromere-associated kinesin)
DE   (MCAK) (Kinesin-like protein 6).
GN   KIF2C OR KNSL6.
OS   Homo sapiens (Human).
OC   Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
OC   Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
OX   NCBI_TaxID=9606;
RN   [1]
RP   SEQUENCE FROM N.A. (ISOFORM 1).
RC   TISSUE=T-cell;
RX   MEDLINE=98094213; PubMed=9434124;
RA   Kim I.-G., Jun D.Y., Sohn U., Kim Y.H.;
RT   "Cloning and expression of human mitotic centromere-associated kinesin
RT   gene.";
RL   Biochim. Biophys. Acta 1359:181-186(1997).
RN   [2]
...
DR   EMBL; U63743; AAC27660.1; -.
DR   HSSP; P17119; 3KAR.
DR   GO; GO:0005871; C:kinesin complex; TAS.
DR   ...
KW   Motor protein; Microtubules; ATP binding; Coiled coil;
KW   Nuclear protein; Alternative splicing.
FT   DOMAIN      1      254      GLOBULAR (POTENTIAL).
FT   ...
SQ   SEQUENCE    725 AA;  81312 MW;  5BDECC133AB4B55C CRC64;
MAMDSSLQAR LFPGLAIKIQ RSNGLIHSAN VRTVNLEKSC VSVEWAEAGGA TKGKEIDFDD
...
```



Aufgabe 4

- Erstellen eines konzeptuellen Modells
 - Erstellen des relationalen Modells
- zur Integrierung der SWISS-PROT Daten

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

```
ID  KF2C_HUMAN  STANDARD;  PRT;  725  ZA.
AC  Q99661;
DT  16-OCT-2001 (Rel. 40, Created)
DT  28-FEB-2003 (Rel. 41, Last sequence update)
DT  15-SEP-2003 (Rel. 42, Last annotation update)
DE  Kinesin-like protein KIF2C (Mitotic centromere
DE  (MCAK) (Kinesin-like protein 6).
GN  KIF2C OR KNSL6.
OS  Homo sapiens (Human).
OC  Eukaryota; Metazoa; Chordata; Craniata; Vertebrata;
OC  Mammalia; Eutheria; Primates; Catarrhini; Hominoidea;
OX  NCBI_TaxID=9606;
RN  [1]
RP  SEQUENCE FROM N.A. (ISOFORM 1).
RC  TISSUE=T-cell;
RX  MEDLINE=98094213; PubMed=9434124;
RA  Kim I.-G., Jun D.Y., Sohn U., Kim Y.H.;
RT  "Cloning and expression of human mitotic centromere
RT  gene.";
RL  Biochim. Biophys. Acta 1359:181-186(1997).
....
```

Eintrag am 29.11.2002

```
ID  MCAK_HUMAN  STANDARD;  PRT;  725  A
AC  Q99661;
DT  16-OCT-2001 (Rel. 40, Created)
DT  15-JUN-2002 (Rel. 41, Last sequence update)
DT  15-JUN-2002 (Rel. 41, Last annotation update)
DE  Mitotic centromere-associated kinesin (MCAK)
GN  KNSL6.
OS  Homo sapiens (Human).
OC  Eukaryota; Metazoa; Chordata; Craniata; Vertebrata;
OC  Mammalia; Eutheria; Primates; Catarrhini; Hominoidea;
OX  NCBI_TaxID=9606;
RN  [1]
RP  SEQUENCE FROM N.A.
RC  TISSUE=T-CELL;
RX  MEDLINE=98094213; PubMed=9434124;
RA  Kim I.-G., Jun D.Y., Sohn U., Kim Y.H.;
RT  "Cloning and expression of human mitotic centromere
RT  gene.";
RL  Biochim. Biophys. Acta 1359:181-186(1997).
....
```



Silke Trißl, Ulf Leser: Molekularbiologische Datenbanken, Übung, SoSe 2003

Versionen im relationalen Modell

- Lineare, tuple-basiert, mit kompletten Versionen
- Anforderungen: Zugriff auf
 - aktuelle Version
 - Zustand der Datenbank zu beliebigem Zeitpunkt d_0
- 2 Varianten
 - Single Table
 - Schattentabellen
- Annahme: Keine Schlüsselwiederverwertung
 - Schlüssel K wird niemals für verschiedene Objekten benutzt
 - Kann z.B. durch Sequenz sichergestellt werden



Silke Trißl, Ulf Leser: Molekularbiologische Datenbanken, Übung, SoSe 2003

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 *
FROM t
WHERE a='T' AND
      k=k0 AND
      d = (SELECT MAX(d)
          FROM t2
          WHERE t2.K=k0)
```

- SELECT alle Tupel zum Zeitpunkt d_0

```
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 !

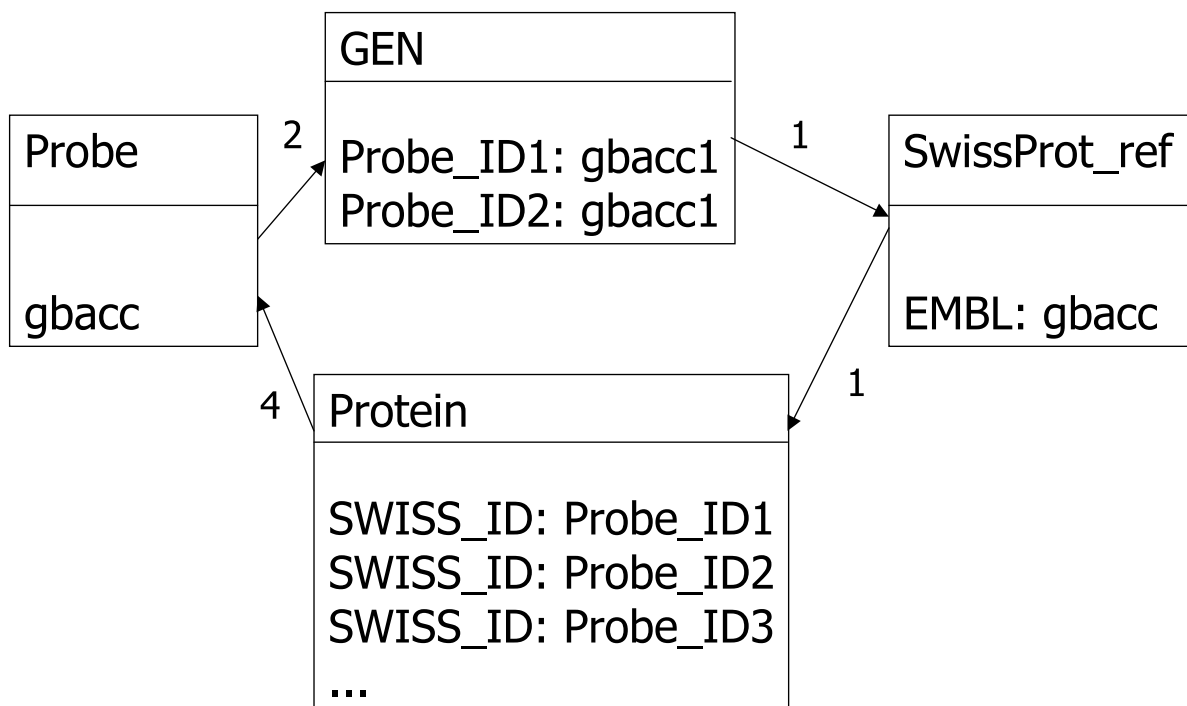
Vergleich

- 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.)
- Nicht behandelt
 - Referenzen / Fremdschlüssel auf versionierte Objekte
 - Identifikation von Änderungen

Queries

- Wie viele der Proben auf dem Array haben einen entsprechenden Eintrag in der aktuellen Version von SWISSPROT?
- Nach wie vielen der Proteine, die in der aktuellen Version von SWISSPROT vertreten sind, können wir auf unseren gespeicherten Arrays Tests durchführen?

Queries – 2 –



Fragen?

- Aufgabe 4 über Goya oder auf Web-page
- Folien auf der Web-page
- Lösung bis 24.06. , 17 Uhr per e-mail oder in RUD25 IV.102

