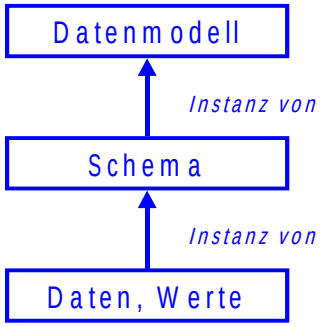
Kapitel 3: Datenmodelle von Bio-Datenbanken

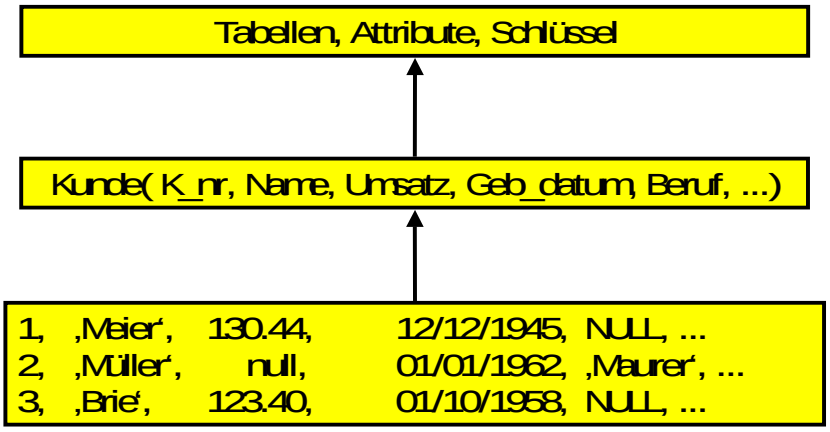
- Grundbegriffe
  - Daten, Schema, Datenmodell
  - Granularität von Datenmodellen
- Datenmodelle
  - Entry-basiertes Datenmodell, ASN.1
  - Relationales Modell
  - Objektorientierte Modelle
  - XML-basierte Modelle
- Spezifische Modellierungsprobleme
  - Objektidentifikation, Versionierung, widersprüchliche Daten

Grundbegriffe

- Daten
  - Tatsächliche Werte, uninterpretiert
  - Ergebnisse von Anfragen
- Schema
  - Beschreibt Typ und Organisation der Werte
  - Spezifiziert durch DDL
- Datenmodell
  - Definition der Modellierungsprimitive, aus denen ein Schema bestehen kann
  - RDMBS: Tabellen, Attribute, ...
  - ORDBMS: Klassen (UDTs), Attribute, Methoden, ...



Beispiel: RDBMS



Granularität eines Datenschemas

- "Breite Datenbanken"
  - "Wenig Klassen (also wenig Detailtiefe), viele Objekte" (d.h. von vielen Objekten wird relativ wenig Information gespeichert)
  - EMBL-Sequenzdatenbank, ArrayExpress, GDB, 2D Page, ...
- "Tiefe Datenbanken"
  - "Viele Klassen (also große Detailtiefe), wenig Objekte" (d.h. von wenigen Objekten wird relativ viel Information gespeichert)
  - Chromosom- / Spezies- / Krankheitsspezifische Datenbanken
- MGD, MIPS, Gencards, ...

Datenmodelle und Systeme

- Entry-basiertes Datenmodell
- Relationales Modell
- Objektorientierte Modelle
- XML-basierte Modelle

Entry-basiertes Datenmodell

- Kein Datenmodell im eigentlichen Sinne
- Flat-File
- Weite Verbreitung in Life Sciences
  - EMBL, Swiss-Prot, Interpro, Omim, Genbank
  - Nahezu alle Bio-Datenbanken haben ein Entry-basiertes Austauschformat
- Beispiel Swiss-Prot
  - Menge von Proteinsequenzen
  - Kernelemente: Sequenz, Taxonomie, Literaturstellen
  - Annotationen: Domänen, Sequenzvarianten, assoziierte Krankheiten, Sekundärstruktur, ...

Entry-basiertes Datenmodell

- Kein Datenmodell im eigentlichen Sinne
- Flat-file
- Weite Verbreitung in Life Sciences
  - EMBL, Swiss-Prot, Interpro, Omim, Genbank ,...
  - Unterstützt von nahezu alle Bio-Datenbanken
- Beispiel Swiss-Prot
  - Menge von Proteinsequenzen
  - Core-Elemente: Sequenz, Taxonomy, Citation
  - Annotationen: Domainen, Sequenzvarianten, assoziierte Krankheiten, Sekundärstruktur, ...

Entries

- Datenbank = Menge ähnlich strukturierter Entries
- Entry: Menge von Feldern (Attribute, Lines),
  - Identifikation durch (standardisierten) Line Code
  - Können 0-n mal vorkommen (semistrukturiert)
  - Können komplexe eigene Struktur haben
  - Können eingebettete Objekte repräsentieren
  - Microsyntax in Werten (Sprechende Schlüssel)
- Keine deklarativen Konsistenzbedingungen
- Kein Klassen- oder Objektbegriff

Line Codes

Line code	Content	Occurrence in an entry
ID	Identification	Once; starts the entry
AC	Accession number(s)	One or more
DT	Date	Three times
DE	Description	One or more
GN	Gene name(s)	Optional
OS	Organism species	One or more
OG	Organelle	Optional
OC	Organism classification	One or more
RN	Reference number	One or more
RP	Reference position	One or more
RC	Reference comment(s)	Optional
RX	Reference cross-reference(s)	Optional
RA	Reference authors	One or more
RT	Reference title	Optional
RL	Reference location	One or more
CC	Comments or notes	Optional
DR	Database cross-references	Optional
KW	Keywords	Optional
FT	Feature table data	Optional
SQ	Sequence header	Once
	(blanks) sequence data	One or more
//	Termination line	Once; ends the entry

Entry-Modell: Ascii-Darstellung

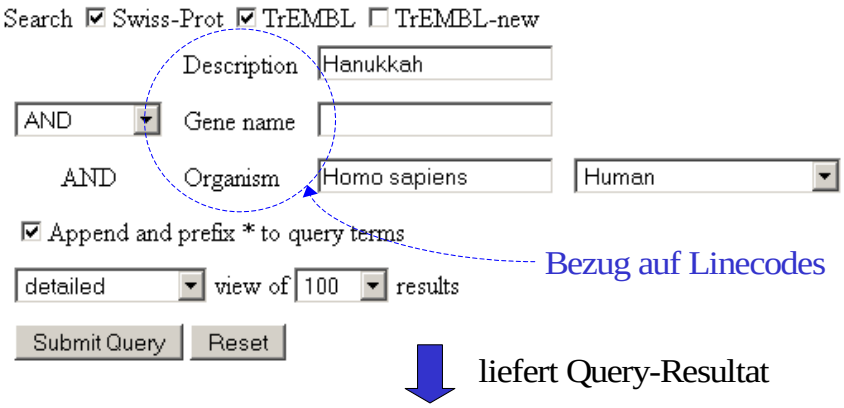
```
ID   GRAA_HUMAN     STANDARD;      PRT;   262 AA.
AC   P12544;
DT   01-OCT-1989 (Rel. 12, Created)
DT   01-OCT-1989 (Rel. 12, Last sequence update)
DT   16-OCT-2001 (Rel. 40, Last annotation update)
DE   Granzyme A precursor (EC 3.4.21.78) (Cytotoxic T-lymphocyte p.
DE   1) (Hanukkah factor) (H factor) (HF) (Granzyme 1) (CTL tryptase)
DE   (Fragmentin 1).
GN   GZMA OR CTLA3 OR HFSP.
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.
RC   TISSUE=T-cell;
RX   MEDLINE=88125000; PubMed=3257574;
RA   Gershenfeld H.K., Hershberger R.J., Shows T.B., Weissman I.L.;
RT   "Cloning and chromosomal assignment of a human cDNA encoding a T
RT   cell- and natural killer cell-specific trypsin-like serine
RT   protease.";
RL   Proc. Natl. Acad. Sci. U.S.A. 85:1184-1
RN   [2]
RP   SEQUENCE OF 29-53.
RX   MEDLINE=88330824; PubMed=3047119;
RA   Poe M., Bennett C.D., Biddison W.E., Blake J.T., Norton G.P.,
RA   Rodkey J.A., Sigal N.H., Turner R.V., Wu J.K., Zweerink H.J.;
RT   "Human cytotoxic lymphocyte tryptase. Its purification from granules
RT   and the characterization of inhibitor and substrate specificity.";
RL   J. Biol. Chem. 263:13215-13222(1988).
RN   [3]
...
```

Feldabhängige Formate (Microsyntax)

Eingebettete Objekte (keine Verweise)

Line codes: Referenz auf (Record-)Struktur einer Zeile (z.B. AC = Accession Code; DT = Date; DE = Description; OS = Organism; OC = Taxonomy)

Entry-Modell: Anfrage (Swiss-Prot)



liefert Query-Resultat

Description and organism

The following is a list of the 1 Swiss-Prot and TrEMBL entries with the description *Hanukkah* and from taxon *Homo sapiens*.

Entries in Swiss-Prot and TrEMBL:

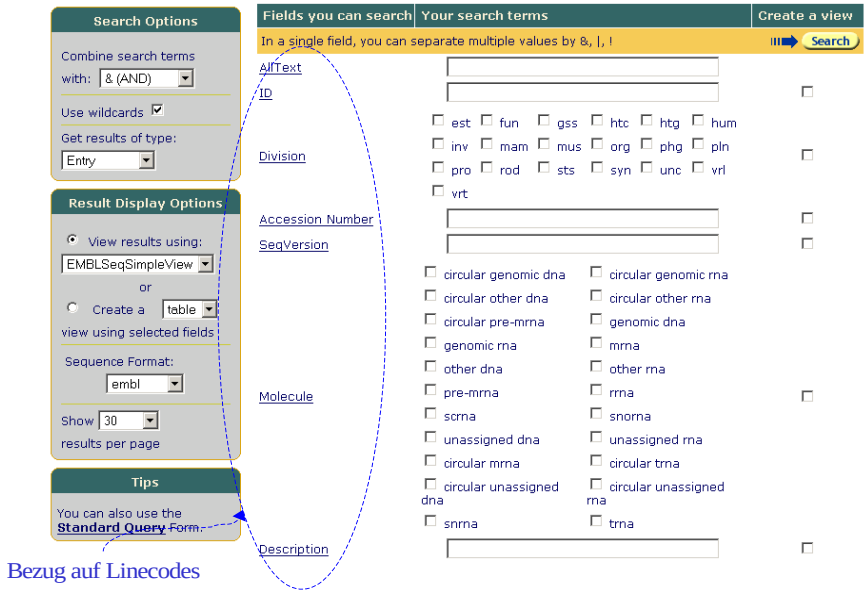
Send selected sequences to Clustal W (multiple alignment) Submit Query Select up to...

Entry name	AC	Gene names	Description	Organisms	Length
MAN	P12544	GZMA, CTLA3, HFSP	**Granzyme A precursor** (EC 3.4.21.78) (Cytotoxic T-lymphocyte proteinase 1) (Hanukkah factor) (H factor) (HF) (Granzyme 1) (CTL tryptase) (Fragmentin 1)	Homo sapiens (Human)	262

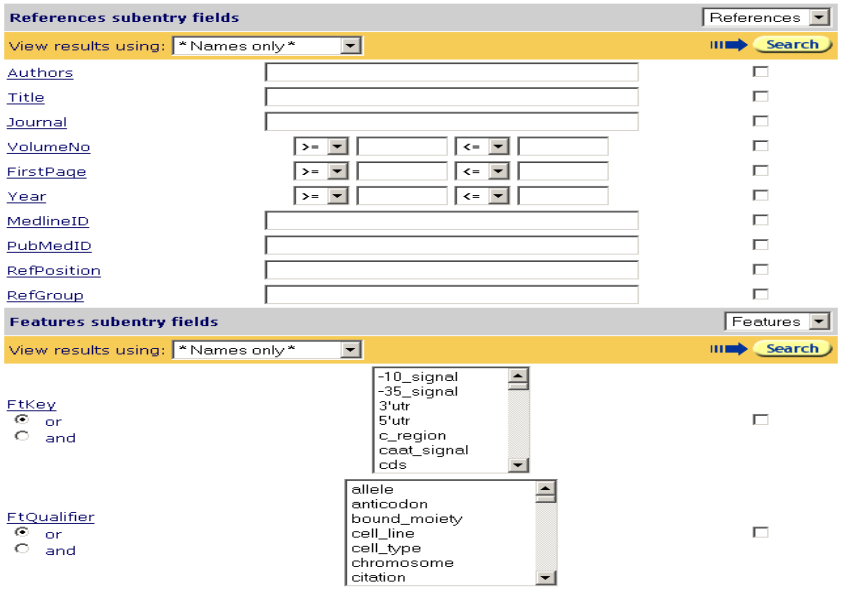
Entry-Modell: Web-Darstellung (Swiss-Prot)

Entry name	**GRAA_HUMAN**
Primary accession number	**P12544**
Secondary accession numbers	None
Entered in Swiss-Prot in	Release 12, October 1989
Sequence was last modified in	Release 12, October 1989
Annotations were last modified in	Release 42, September 2003
Name and origin of the protein	
Protein name	**Granzyme A [Precursor]**
Synonyms	**EC** 3.4.21.78 **Cytotoxic T-lymphocyte proteinase 1** **Hanukkah factor** **H factor** **HF** **Granzyme 1** **CTL tryptase** **Fragmentin 1**
Gene name	**GZMA** or **CTLA3** or **HFSP**
From	Homo sapiens (Human) [TaxID: 9606]
Taxonomy	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
References	
[1] SEQUENCE FROM NUCLEIC ACID. **TISSUE**=T-cell; MEDLINE=88125000; PubMed=3257574; [NCBI, ExPASy, EBI, Israel, Japan] Gershenfeld H.K., Hershberger R.J., Shows T.B., Weissman I.L.; "Cloning and chromosomal assignment of a human cDNA encoding a T cell- and natural killer cell-specific trypsin-like serine protease."; Proc. Natl. Acad. Sci. U.S.A. 85:1184-1188(1988).	

Entry-Modell: Anfragemasken (EMBL)



Entry-Modell: Anfragemasken (EMBL) (2)



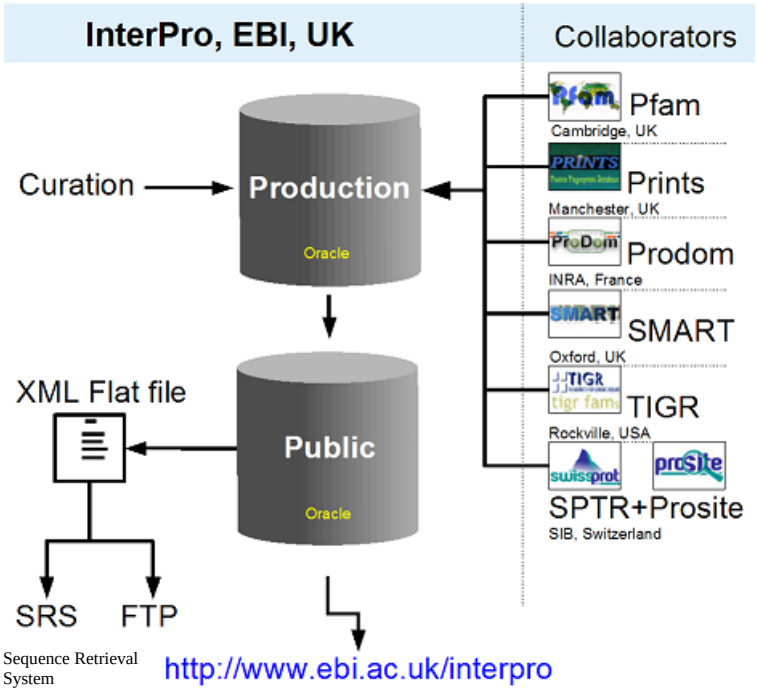
Entry-Modell: Zusammenfassung

- Datenmodell
  - Einziges Modellierungsprimitiv: Der Entry
  - Felder mit hierarchischer Schachtelung; Keine Assoziationen
- Schema: Keine explizite Repräsentation auf Basis einer DDL
- Werte: Einfache und zusammengesetzte Werte möglich
- Eher Format als Datenmodell
- Vorteile
  - Sofort lesbar für Menschen, plattformunabhängig (ASCII), hohe Flexibilität durch textorientieres Editieren, leicht zu durchsuchen (Grep, "search"-Button)
- Nachteile
  - Keine Konsistenzbedingungen
  - Hohe Redundanz geschachtelter Objekte: Literatur, Taxonomie, ...
  - Keine strukturierten Anfragen möglich

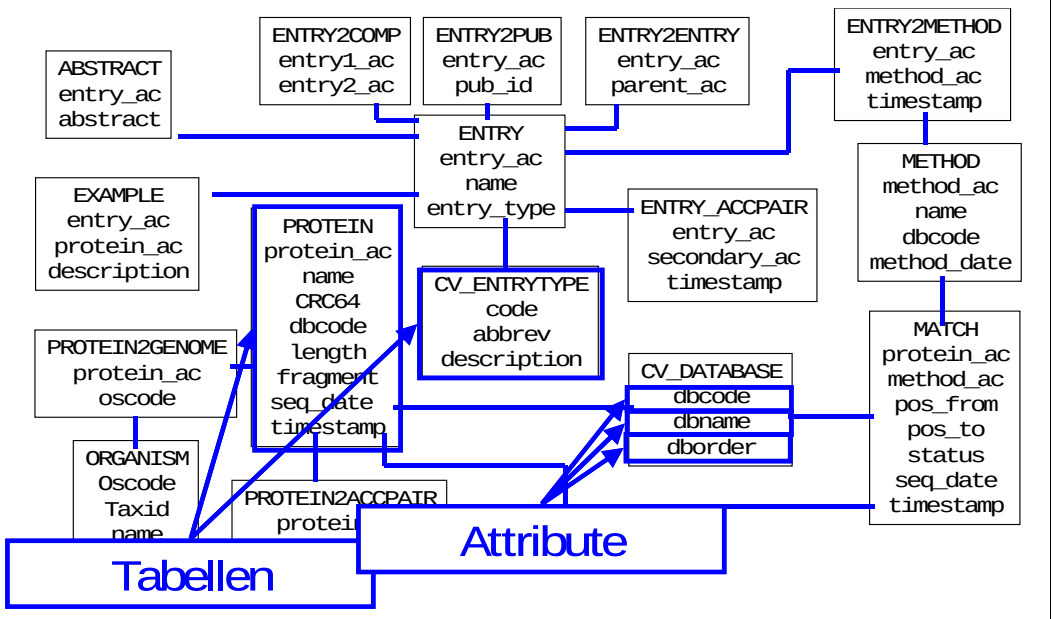
Relationales Modell

- Industriestandard
- Konzentration auf Speicherung/Retrieval
  - Semantisch arm, wenig Elemente
  - Nicht als Designmodell gedacht (wie z.B. ER oder UML)
  - Userinterface müssen programmiert werden
- Entwickelt für Transaction-Processing, Mehrbenutzerbetrieb, Client-Server
  - Overhead für typische "Read-Only" Bio-DB‘s
  - Komplizierte Installation, Administration, Backup, ...

Beispiel: InterPro



Interpro: Relationales Schema (Auszug)



Relationales Modell: Zusammenfasssung

- Datenspeicherung und -retrieval
- Vorteile
  - Strukturierte Anfragen
  - Sehr weit verbreitet, robust, Industriestandard
  - Skalierbarkeit und Optimierbarkeit
  - Viele Produkte verfügbar
  - Ständige Weiterentwicklung
- Nachteile
  - SQL schwierig zu lernen
  - Volltextsuche nicht direkt möglich
  - Datenaustausch zw. Forschungsgruppen schwieriger als mit Entry-basierten Flatfiles

Abbildung Entry-basiert - Relational

- Flatfiles
  - Mengenwertige Attribute
  - Geschachtelte Attribute, hierarchisch, keine Beziehungen
- Natürliche Abbildung

```
FUNCTION entryToRelation( E)
  erzeuge relation R
  erzeuge Surrogateschlüssel K zu R
  FORALL A aus E               // Attribute
    CASE
      A einwertig -> A wird neues Attribut von R
      A mehrwertig -> erzeuge Relation A' mit FK K
      A komplex -> erzeuge Relation A' mit FK K
                   A' = entryToRelation( A)
    END CASE
  END FORALL
END
```

Entscheidungsspielräume

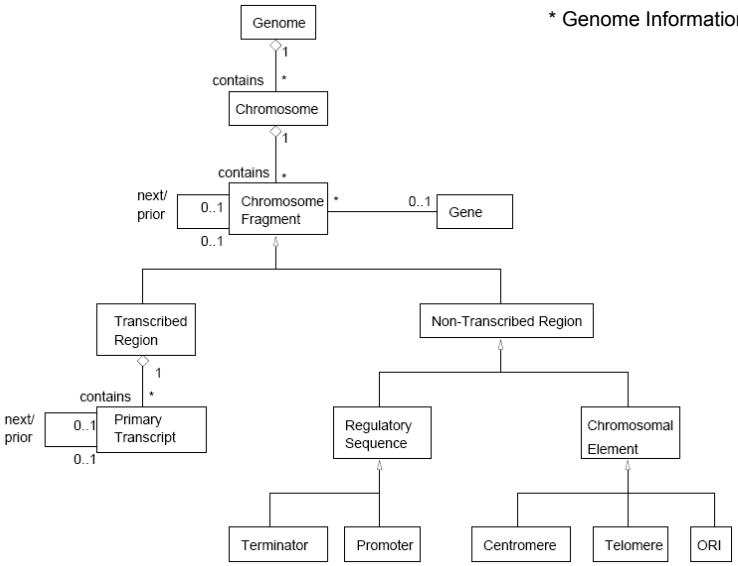
- Ist ein Attribut einwertig/ mehrwertig/ komplex?
  - Muss festgelegt werden
  - Attributparser i.A. nicht ableitbar
- Microsyntax: Auflösen oder belassen?
  - Erste Normalform
- Ordnung kann, muss aber nicht wichtig sein
  - Beispiel: Taxonomy
  - Gegenbeispiel: Publikationen
  - Abbildung in „Rank“ Attribute der Relation A‘
- Redundanz: Zusammenfassen doppelter Subentries?
  - Beispiel: Publikationen
  - Erfordert (schwieriges) Testen auf Gleichheit

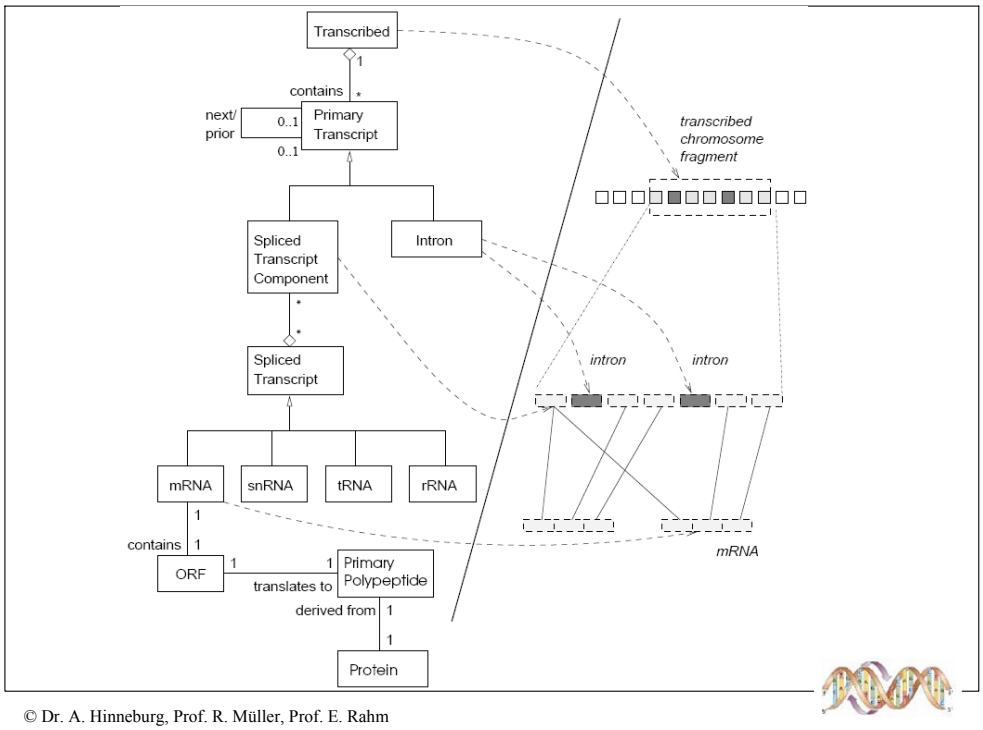
Objektorientierte Modelle

- UML, OPM (Object-Protocol Model), ACeDB
- UML: Industriestandard zur Modellierung von
  - Software: Klassen-Diagramme, Sequenzcharts, Zustands-Diagramme
  - Architekturen: Verteilungs-Diagramme, Komponenten-Diagramme
  - Requirements: Use Cases
  - Prozessen: Aktivitäts-Diagramme, Collaboration-Diagramme
- Viele UML-Tools: Rational, TogetherJ, Argo-UML, ...
- DB-Design mit UML: Klassendiagramme
  - Modellierung in UML mit späterer Übersetzung in relationale Schemata
  - Beispiele: SP, EMBL, GIMS, ArrayExpress, ...
  - Direkte Umsetzung in ORDBMS möglich

UML-Beispiel GIMS*

* Genome Information Management System



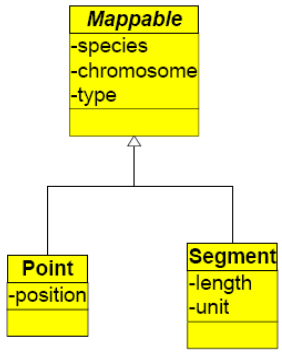


UML: Zusammenfassung

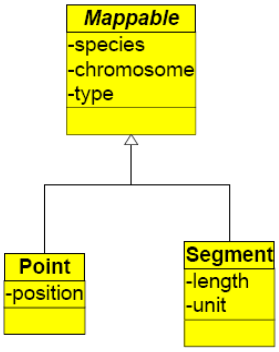
- Vorteile
  - Reichhaltiges Datenmodell mit klar definierter graphischer Notation (durch Metamodell)
  - Industriestandard für Modellierung / Entwicklung
  - Enge Verkopplung mit Software möglich (Automatische Erzeugung von Persistenzschicht: Schema plus Klassen)
- Nachteile
  - Dualität OO – RDBMS nicht trivial
  - OO-Übersetzung erzeugt wenig intuitive Schema
  - Keine Anfragesprache definiert
  - Keine Untersützung von semistrukturierten Daten

Objektrelationales Mapping

- Vier Varianten
  - Union: 1 Tabelle
  - Vertikale Zerlegung: 1 schmale Tabelle pro Klasse
  - Horizontale Zerlegung: 1 breite Tabelle pro Klasse
  - Volle Materialisierung
- Fragen
  - Speicherverbrauch: Redundant
  - Extension: Zugriff auf alle Objekte einer Klasse
  - Intension: Zugriff auf alle Attribute einer Klasse



Union Tabelle



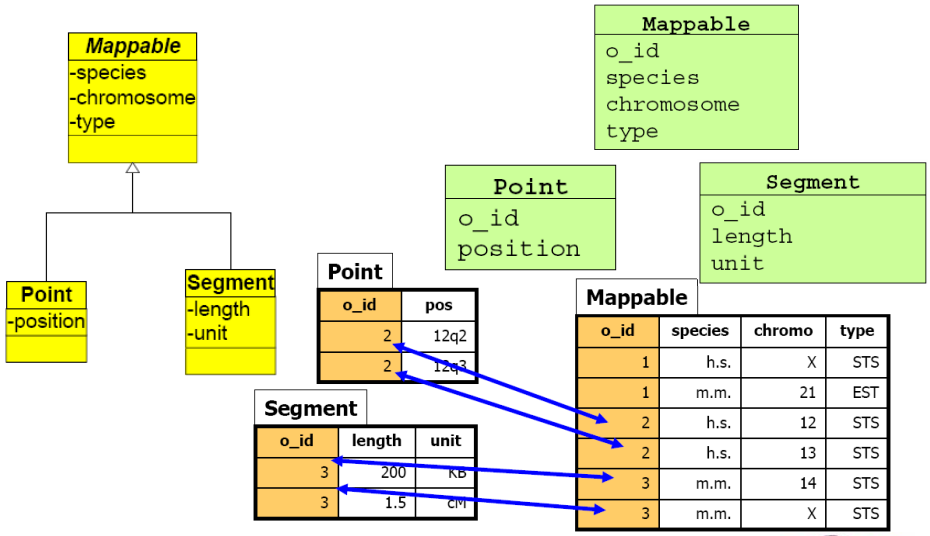
MapHierarchy

```
class_id
species
chromosome
type
length
unit
position
```

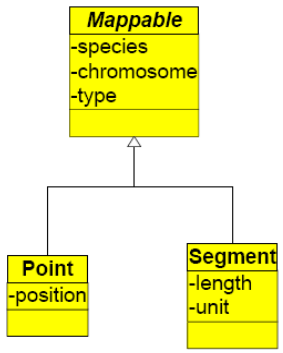
MapHierarchy

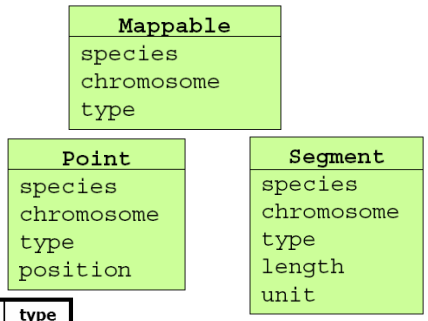
class_id	species	chromo	type	length	unit	pos
1	h.s.	X	STS			
1	m.m.	21	EST			
2	h.s.	12	STS			12q2
2	h.s.	13	STS			12q3
3	m.m.	14	STS	200	KB	
3	m.m.	X	STS	1.5	cM	

Vertikale Zerlegung



Horizontale Zerlegung





Mappable

species	chromo	type
h.s.	X	STS
m.m.	21	EST

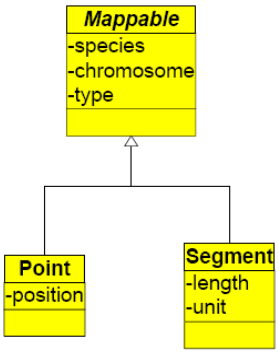
Point

species	chromo	type	pos
h.s.	12	STS	12q2
h.s.	13	STS	12q3

Segment

species	chromo	type	length	unit
m.m.	14	STS	200	KB
m.m.	X	STS	1.5	cM

OR Volle Materialisierung



Point
species
chromosome
type
position

Mappable
species
chromosome
type

Segment
species
chromosome
type
length
unit

Mappable

species	chromo	type
h.s.	X	STS
m.m.	21	EST
h.s.	12	STS
h.s.	13	STS
m.m.	14	STS
m.m.	X	STS

Point

species	chromo	type	pos
h.s.	12	STS	12q2
h.s.	13	STS	12q3

Segment

species	chromo	type	length	unit
m.m.	14	STS	200	KB
m.m.	X	STS	1.5	cM

Vergleich

	Speicher-verbrauch	Zugriff Extension	Zugriff Intension	Konsistenz-sicherung	Inserts / Updates
Union	Hoch	Eine Query; Bedingung auf class_id	Eine Query; Bedingung auf class_id	NULL-Werte garantieren	1 Insert
Vertikal (Schmal)	Gering (OID doppelt)	Eine Query (nur OID + wenige Attribute)	N Joins (Alle Vorfahren)	FK von Kindertabs dürfen nicht überlappen	N Insert (Alle Vorfahren)
Horizont. (Breit)	Minimal	M Unions (Alle Nachfahren)	Eine Query		1 Insert
Voll	Hoch	Eine Query	Eine Query	Redundante Daten – Anomalien	N Insert (Alle Vorfahren)

Bewertung

	Speicher-verbrauch	Zugriff Extension	Zugriff Intension	Konsistenz-sicherung	Inserts / Updates
Union	Hoch	Eine Query	Eine Query	NULL-Werte garantieren	1 Insert
Vertikal (Schmal)	Gering	Eine Query	N Joins	Überlappende FKs	N Insert
Horizont. (Breit)	Minimal	M Unions	Eine Query		1 Insert
Voll	Hoch	Eine Query	Eine Query	Anomalien	N Insert

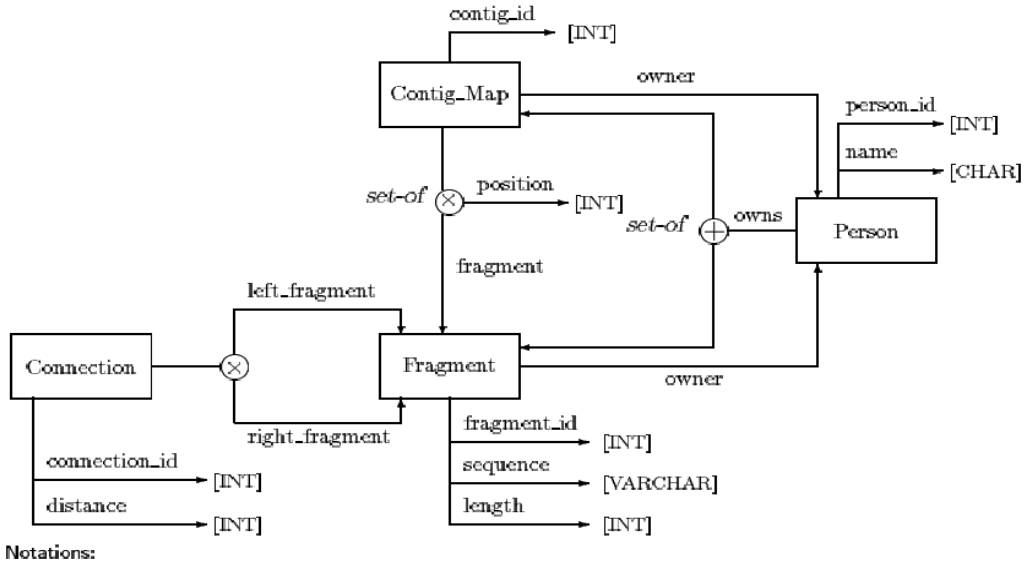
- => Optimale Methode ist anwendungsabhängig

OPM

- Object Protocol Model (OPM) für Repräsentierung wissenschaftlicher Experimente (Chen et al. 1995)
- Objektorientiertes Datenmodell
  - Vergleichbar mit ODMG
  - Erweiterungen für (Bio-)Wissenschaften
- Gewisse Verbreitung in Life Science
  - GSDB*, GDB, MGD, ...
- OPM-QL und OPM*QL

* Genome Sequence DataBase

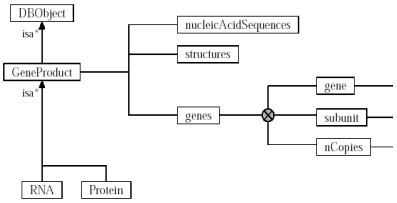
OPM: Graphische Syntax



Notations:

- Labeled rectangles represent object classes.
- Arrows represent attributes, with the arrow pointing to the value class associated with the attribute.
- Circles containing a "+" denote unions of value classes.
- Circles containing a "×" denote tuple attributes.

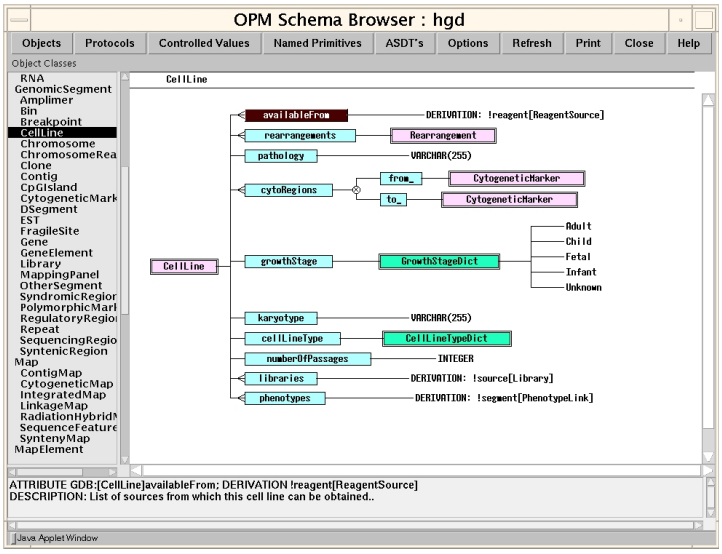
OPM: Beispiel GDB




```
OBJECT CLASS GeneProduct  isa* DBObject
 DESCRIPTION: "Proteins or RNAs resulting from the
                transcription of this gene."
 PROPERTIES: "FE_Editable" "No"
 ATTRIBUTE  nucleicAcidSequences
      DERIVATION: ! dBObject [ NucleicAcidSequenceLink ]
            ORDER BY nucleicAcidSequences.displayName ASC
      DESCRIPTION: "External links to nucleic acid sequence databases."
            PROPERTIES: "FE_QueryHelp" "<a href={{uptodb}}/?!action=help&type=query#RefExt
             target=gdbHelpWindow>Tips for Searching attributes that link outside GDB</a>"
            ...
 ATTRIBUTE structures
      DERIVATION: ! geneProduct [ StructureLink ] ORDER BY structures.displayName ASC
      DESCRIPTION: "External links to structure databases."
            PROPERTIES:
                  ...
 ATTRIBUTE genes (gene, subunit, nCopies):
      set-of [1,] ([1,1] Gene, [0,1] VARCHAR(20), [0,1] SMALLINT)
      DESCRIPTION: "List genes coding for this product. ... "
            PROPERTIES: "OB_Key" "gene", "GDB_Rule" the releasedate of the GeneProduct cannot be any
                  earlier than the max release date of the genes"
            COMPONENT gene
                  PROPERTIES:
                  "opm_delete" "tuplerestricts"
                  …
```

OPM: Toolbox

- Graphical Schema Editor
- Retrofitting Tool (View Based)
- Schema Translator
- Query Translator
- Webinterface Creator
- Doc Creator

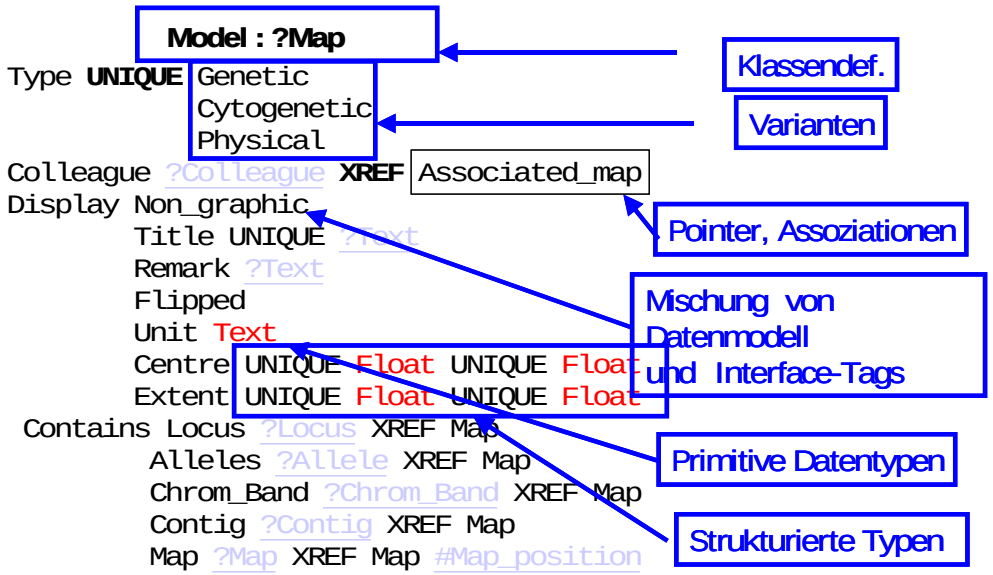


ACeDB*

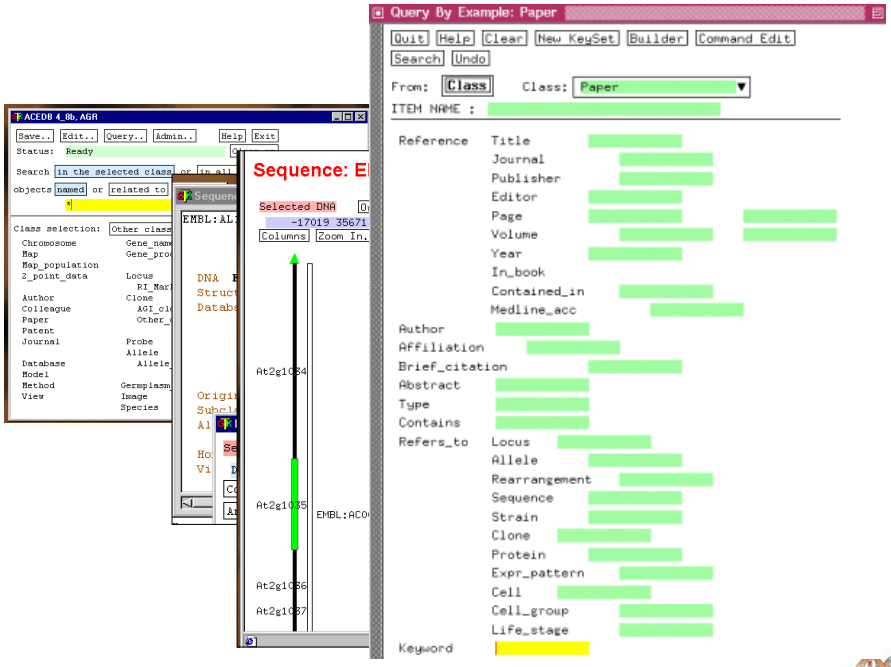
- DBMS, ursprünglich entwickelt zur Abbildung des Genoms von Caenorhabditis elegans (von und für Biologen)
- Weitere Verwendung für Genom vieler anderer Spezies (auch im Rahmen des HGP)
- Tools für
  - Storage, Anfragen, GUI, Webserver etc.
- Verfügbar auf vielen Plattformen (Mac, Linux, Windows)
- Datenmodell
  - "Schwach" objektorientiert
  - Semistrukturiert

* A C. (Caenorhabditis) elegans DataBase (Caenorhabditis elegans: eine Nematoden-Art; Nematoden = Rundwürmer)

ACeDB: Datenmodellierung



ACeDB: Screenshots

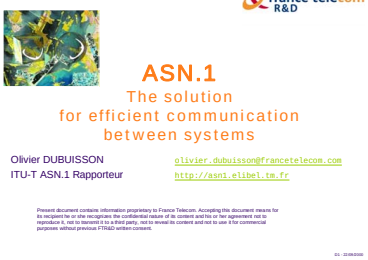


ACeDB: Zusammenfassung

- Unbeliebt bei Informatikern, beliebt bei Biologen
- Ausgerichtet auf manuelles Editieren und Browsen
  - Queries sind schwierig (auch für Biologen) und langsam
  - „Archive“: Schreiben darf nur einer
  - Keine API für Updates, Inserts, Deletes
  - ACE Files
  - Unklare Semantik
- Viele Gerüchte
  - Skalierbarkeit jenseits 1GB ?
  - Fehler in Datenmodell (Tabs !) – Crash
  - Recovery nach Crash ?
  - Dennoch wg. praktischer Relevanz und derzeitiger Verbreitung nicht zu unterschätzen!

ASN.1

- Abstract Syntax Notation One (http://asn1.elibel.tm.fr/)
- Formatbeschreibungssprache ähnlich DTD / XML Schema
- Ursprünglich für Definition von Datenaustauschformaten in der Telekommunikationsbranche
- Internationaler Standard (1984) ISO 8824 / 8825
- Verwendet von NCBI*-Datenbanken
  - Genbank
  - UniGene
  - dbSNP
  - ...



ASN.1: Elemente

- Datenmodell mit expliziten Typen (im Gegensatz z.B. zum Entry-Modell)
- Ein Typ besteht aus
  - Primitiven Attributen
  - Strukturen (structs)
  - Sequenzen (mengenwertige Attribute)
  - Choices (variants)
  - Constraints, z.B. Lottery-number ::= INTEGER (1..49)

```
Married ::= BOOLEAN
Age ::= INTEGER
Picture ::= BIT STRING
Form ::= SEQUENCE { name                PrintableString,
                    age                 Age,
                    married             Married,
                    marriage-certificate Picture OPTIONAL }
```

```
Quantity ::= CHOICE { units       INTEGER,
                      millimeters INTEGER,
                      milligrams  INTEGER }
```

- ASN.1-Schema: ASCII-Definition von Typen
- Language Mappings verfügbar (für Java, C++, C, COBOL, XML, Perl)
- XML-Mapping von NCBI-Datenbanken verwendet (ASN2XML)

* National Center for Biotechnology Information

ASN.1: Beispiel

```
NCBI-PubMed DEFINITIONS ::=
BEGIN

EXPORTS Pubmed-entry, Pubmed-url;
IMPORTS PubMedId FROM NCBI-Biblio
        Medline-entry FROM NCBI-Medline;

Pubmed-entry ::= SEQUENCE {        -- a PubMed entry
    -- PUBMED records must include the PubMedId
    pmid PubMedId,
    -- Medline entry information
    medent Medline-entry OPTIONAL,
    -- Publisher name
    publisher VisibleString OPTIONAL,
    -- List of URL to publisher cite
    urls SET OF Pubmed-url OPTIONAL,
    -- Publisher's article identifier
    pubid VisibleString OPTIONAL
}

Pubmed-url ::= SEQUENCE {
    location VisibleString OPTIONAL, -- Location code
    url VisibleString                -- Selected URL for location
}

END
```

Modularer Aufbau

Export / Import von Modulen

Primitive / komplexe Datentypen

ASN.1: Beispiel aus NCBI

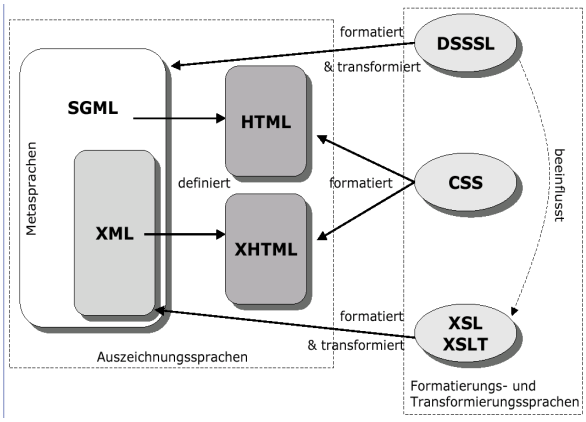
```
Seq-entry ::= set {
  level 1 ,
  class nuc-prot ,
  release "" ,
  descr {
    source {
      org {
        taxname "Adeno-associated virus 2" ,
        db {
          {
            db "taxon" ,
            tag
              id 10804 } } ,
        orgname {
          name
            virus "Adeno-associated virus 2" ,
          lineage "Viruses; ssDNA viruses;
                   Parvoviridae;
                   Parvovirinae;
 Dependovirus" ,
          gcode 1 ,
      inst {
        repr raw ,
        mol dna ,
        length 4675 ,
        strand ss ,
        seq-data
          ncbi2na 'FA51D5DDE6676767478A5A98502B656195A9FE56A6974B89898999222A2
E941D4D1CAAF5E8AAE8B6E1B83C6D32AF2A2B5ECF22B46E2EFF984FF98453BAD19EACFC2562E24
64AB753FE09A8AFE0664965396ABFC623EE3C2B552617E1A93795A4FDE127FB81EAE9620AOEA2F
96523DE13A378378F892915785BA5882792661FDE1AOE996EE2C29568A57FDFEE43F882A22271F
51391B9DBA0145AAE0353AFFA86FD78B48F66001E3D220FC59A8D8961FE501EBD9AD1021480E99
68A6A042BAE8E2E713550F1F9D5401525E2752EA6E870CE812CFC265EFEODD1A26C06BEBA64937
8646ED92192892040883483543DE3996B8D2340O7D252B13A27ADABA76E842A8F1768824BA3528
A14A5D31375F439A5D41D9AD503429E5FA10E6AO23CE25E1C0165561C5EBA949256E8A13F52436
8FCCO3FE81CO6AC63550CE69F5B7F7A8EA5180OBDA4228114DE9EFEA5E41C5A821413668A53254
47B95F71AB9BO1E850E207F57DO61EEDB423AE37AEA28AAO8E1650ADBA2DA5O253DDA28242B99B
A1480E42D76952321587S6E36D17504504EE65B8F86AO74185F60452496F9085A3BDO3F81D1596
DE8D387FA82B4509282DO21FFD6BA9028D1BAF8AE89383DC6DO02AE8942O0856554B8648CC2E25
406AE662D2F992536lB4866827D8DO719212B140103BDDB46EA4EODE39EFD5E484398883834830
03379F4746848087BF22E7F56ED2ODDO56FDEDB400A6CD201EE713D34CD3A8OAE5219F91E5E637
ADOEE8FE8E1E4DFEO43O38FCOD2B3A7963AF37D48FA76284777782830B4BAE8274O5E951451409
59226930A18492AB7EE7D7AB10B176857D0687610A8896B418A48659A576246C4097185A49D849
A21056C5DOB10519619A2FD28997C08231B7FEAA41768624B7D4A6O022AF7EO5DEA5EBE2817BC2
1A75A8OO2296B224775EE894875D76AO5AO29A94925E420023EOFFAD21E8864874B17855497768
494524955DEB7A81C318E9C4A4BA6450E9210C18A9961A2EAC3D75AO3E93E63D44E8EA6122D345
14915817A95E545C410517710103F525OD289760610D1C7FA712455FAAB3FE1F4123D47947FD14
6E1E9021D341041EA8F5854221DO7DO9DFC13DOB408AD1920E1AC61863E50C17C5246BD2BBF1E1
DA2C52756C6D769DA64DOA397565BD52486DF4EB944B3A3174578106A2D292CA19DF4FF1E5E8B1
FD7DD239E6C5AO107F17D2711FF8A1BD7F51249C6744948B7A16DD3835DD3614B17B3C7E248101
1D42E8145192DOA7D2FF74A5A262E13DA852DCAO7A7D7A15EF1652498B34084DE68C10412E0C76
E87A27142C51743A488777AEOD694E90944286382OOBFFD7489AAF74DFEAO90A7488O103B813E0
0AD38F12182280DAO414356E9C6892CEBDECDC505D488A41210927164BEDO44429BDF5293ADE92
8488EEC5F4AA54DEA408F5111A1A13FD157755D3ABA3DA1FO115D75123DD3420455AC5E60D7D85
17D2E6902FE7D73444B1D46A11AD26E88D8BA89E48280124O67AOD58OF4B11F5071042DEFODBA
1F16E8C7OE9BB3D225D954FA452317876C37BOF9FBC343016FC3DBF4BEO7FADDE6CFDFDF372FD4
E9C6C8C2C93A6AFOD3C1C4281572E3A2FA51D5DDE6676767478A5A98502B656195A9FE56A6974B
89898999222A2E940'H } ,
      annot {
        {
          data
```

ASN.1: Zusammenfassung

- Vorteile
  - Binäres Encoding sehr kompakt (1 Base – 2 Bit)
  - Vollständige Toolbox erhältlich (NCBI)
  - Plattformunabhängig
- Nachteile
  - Wenig verbreitet
  - Binäres encoding für Menschen unlesbar
  - Text Encoding schwierig zu parsen
- Zukunft in Life Science unklar
- Vermutlich Ablösung durch XML

Kapitel 3 (Forts.): XML-basierte Modelle, Spezielle Modellierungsaspekte

- XML: Extensible Markup Language
  - Version 1: 1998
  - Einschränkung von SGML, Erweiterung von HTML
  - W3C-Standard
  - Kern einer Sprachgruppe: XSL, Xpath, XQuery, XLink, ...



- Standard zur Definition von Austauschformaten

```
<RefAuthors>
   Moore W.S., DeFilippis V.R.
</RefAuthors>
<RefTitle>
   The window of taxonomic resolution ...
</RefTitle>
```

XML, DTD und XML Schema

- Document Type Definition (DTD)
  - Definition erlaubter Elemente und ihrer Attribute
  - Assoziationen durch ID, IDREF
  - Unzureichende Datentypisierung
- XML-Schema: Erweiterung
  - Constraints und Kardinalitäten
  - Vordefinierte und benutzerdefinierte Datentypen
  - Einfache und komplexe Datentypen
- XML-Dokument ist
  - Wohlgeformt: Entspricht XML Syntax
  - Gültig/valide bzgl. einer DTD: Entspricht einer gegebenen DTD
- Speicherung
  - Flatfile, XML-Datenbank (Tamino, ...), Relationales Mapping

XML in der Bioinformatik

- BIOML (BIOploymer Markup Language)
  - Information über Experimente mit Proteinen, Genen, ...
  - Verbindet pysikalisches Objekt mit Experimentaldaten
  - http://www.bioml.com//BIOML/
- BSML (Bioinformatic Sequence Markup Language)
  - Daten-Modell für Bezüge zwischen Sequenzen und Phenomenen auf verschiedenen Ebenen (molekular bis Genomebene)
  - http://www.labbook.com
- PSDML (Protein Sequence Database Markup Language)
  - Austausch für PIR, http://pir.georgetown.edu/
- GAME (Genome Annotation Markup Elements)
  - Informationen über Sequenzabschnitte, (automatisch oder manuell)
  - http://www.bioxml.org/Projects/game/game.dtd.html

XML in der Bioinformatik

- SBML (Systems Biology Markup Language)
  - Beschreibung für Modelle (Pathways), http://www.sbml.org
- CellML
  - Computerbasierte biologische Modelle, http://www.cellml.org
- MAGE-ML (Microarray Gene Expression)
  - Informationen über Genexpressions-Experimente
    - Design,
    - Hersteller Infos,
    - Experiment Setup und Ausführung
    - Genexpressionsdaten
    - Datenanalyse
  - http://www.mged.org/Workgroups/MAGE/

Elemente und Attribute

- Zwei Arten um Daten zu kapseln
- Attribute
  - Meist 1:1 Beziehung zum Datenelement, z.B. ID
  - Metainformationen, z.B. Maßeinheiten
- Elemente
  - Normale Syntax mit Start und Ende, <gene>CD4</gene>
  - Leere Syntax, keine gekapselten Daten außer Attribute
    - <db_xref db="EC" dbkey="2.7.4.0"/>
    - Effiziente Art Attribute zu speichern
  - Strukturelemente ohne Daten -> hierarchische Informationen
  - Leere Elemente als Begrenzer für Listen
  - Kürzel: &spdb kann für SwissProtRelease stehen
  - Sonderzeichen wie < > " ' & beachten

Probleme mit XML

- Allgemeine Elemente als Unterelement mit unterschiedlicher Semantik
  - z.B. BSML: Element Attribute für Versionen, Quellen und Organismen
    - <Attibute name="version" content="AB003468.1"/>
    - <Attibute name="source" content="cloning vector pAP3neo DNA"/>
    - <Attibute name="organism" content="cloning vector pAP3neo"/>
- Syntaktische Probleme
  - Special Character in Daten
  - Datentyp ist abhängig von anderem Attributewert (Metadaten)
  - Objektreferenzen müssen ersetzt werden, wenn sie XML Markup enthalten
  - Datenfeld kann Listen enthalten
- Strukturprobleme
  - sehr viele Strukturelemente
  - stark verschachtelte Struktur
  - Datenobjekt ist auf mehrere Dateien verteilt
  - Redundante Einträge

GAME

- DTD + Tools für Austausch von Genom-Annotationen
- Ergänzt GFF*-Format
  - GFF: standardisiert Darstellung der Genstrukturen
  - GAME erweitert GFF mit Metainformationen (Annotationen)
- Überschneidung mit OMG-LSR† BSA‡
- CORBA als Interface, Daten in XML

GAME Semantics

- Annotation
  - *"A collection of features found on an associated set of sequences"*
- Features
  - *"Conclusions describing intervals on different sequences. Supported by analytical evidence"*
- Analyses
  - *"Computer or biological experiments on a sequence. Results apply to sequence interval"*
- Sequences
  - *"Biological sequences in which we're interested"*

* Genefinding File Format: Format zur Abspeicherung von Genstrukturen
† Life Science Research Domain Task Force der OMG
‡ Biomolecular Sequence Analysis

GAME: DTD-Ausschnitt

```
<!ELEMENT game ANY>
...
<!ELEMENT offset (#PCDATA)>
<!ELEMENT length (#PCDATA)>

<!ENTITY % site_operator
  " site_operator (less_than | greater_than)">

<!ELEMENT fuzzy_start (span)>
  <!ATTLIST fuzzy_start
            %site_operator; #IMPLIED>

<!ELEMENT fuzzy_end (span)>
  <!ATTLIST fuzzy_end
            %site_operator; #IMPLIED>

<!ELEMENT fuzzy_span (fuzzy_start, fuzzy_end)>

<!ELEMENT span (offset, length)>
  <!ATTLIST span
            between    (TRUE) #IMPLIED
            either_dir (TRUE) #IMPLIED>
```

Location: 340..565

```
<span>
   <offset>339</offset>
   <length>225</length>
</span>
```

Location: <345..500

```
<fuzzy_span>
   <fuzzy_start site_operator="less_than">
      <span>
         <offset>344</offset>
         <length>1</length>
      </span>
   </fuzzy_start>
   <fuzzy_end>
      <span>
         <offset>499</offset>
         <length>1</length>
      </span>
   </fuzzy_end>
</fuzzy_span>
```

GAME: Pfam-Beispiel

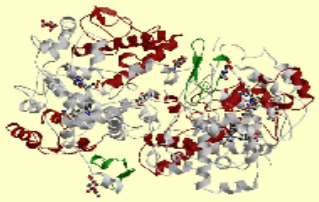


Figure 1: 6cox
Oxidoreductase
Cyclooxygenase-2 (prostaglandin synthase-2) complexed with a selective inhibitor, sc-558 in i222 space group

Key:

Domain	Chain	Start Residue	End Residue
An_peroxidase	A	228	344
EGF	A	36	69
An_peroxidase	A	464	521
An_peroxidase	B	228	344
EGF	B	36	69
An_peroxidase	B	464	521

Beispiel für Protein-Domain: EGF-like domain
(Pfam-ID PF00008)

A sequence of about thirty to forty amino-acid residues long found in the sequence of epidermal growth factor (EGF) has been shown PUB00001077, PUB00001077, [MEDLINE:84117505], [MEDLINE:91145344], [MEDLINE:85063790], PUB00004964 to be present, in a more or less conserved form, in a large number of other, mostly animal proteins.
The list of proteins currently known to contain one or more copies of an EGF-like pattern is large and varied. The functional significance of EGF domains in what appear to be unrelated proteins is not yet clear. However, a common feature is that these repeats are found in the extracellular domain of membrane-bound proteins or in proteins known to be secreted (exception: prostaglandin G/H synthase). The EGF domain includes six cysteine residues which have been shown (in EGF) to be involved in disulphide bonds. The main structure is a two-stranded -sheet followed by a loop to a C-terminal short two-stranded sheet. Subdomains between the conserved cysteines vary in length.

GAME: XML-Darstellung von Pf00008

```
<computational_analysis seq="dmNotch">
<date>08/26/1999</date> <program>hmmpfam</program> <version>2.1.1</version>
<database>
          <name>Pfam</name> // Bezug auf “Pfam” (Protein families database of
                            // alignments and HMMs
          <date>god (and Sean Eddy) knows when it was created</date>
          <version>4.1</version>
</database>
<result_set>
          <dbxref>
          <database> <name>Pfam</name> </database>
          <unique_id>PF00008</unique_id> // Pfam accession number
          </dbxref>
          <output> <type>Description</type><value>EGF-like domain</value> </output>
          …
          <result_span>
             <score> 22.6 </score>
             <type>Motif</type>
             <subtype>EGF</subtype> // EGF: epidermal growth factor (family)
             <seq_relationship seq="dmNotch" type="query">
                    <span>
                       <offset>62</offset>
                       <length>32</length>
                    </span>
                    <alignment>CTSV-GCQNGGTCVTQLN------GKTYCACDSH-----YVGDYC</alignment>
             </seq_relationship>
             <seq_relationship seq="EGF" type="subject">
                    <span>
                       <offset>0</offset>
                       <length>44</length>
                    </span>
                    <alignment>CapnnpCsngGtCvntpggssdnfggytCeCppGdyylsytGkrC</alignment>
             </seq_relationship>
           </result_span>
</result_set>
</computational_analysis>
```

XML in der Bioinformatik: Bewertung

- Gut geeignet für semi-strukturierte Biodaten
- Vorteile (insb. gegenüber Entry-basiertem Modell)
  - Industriestandard, viele Tools (Editoren)
  - DTD generierbar aus UML-, Java-Spezifikationen, ...
  - Effiziente Parser
  - Unterstützung durch relationale DB-Hersteller (IBM, Oracle, ...)
  - Zunehmend XML-Datenbanken verfügbar (Tamino etc.)
  - Strukturierte Anfragen (XQuery) und Textsuche möglich
- Nachteile
  - Dokumente sehr lang, daher oft nicht sehr gut lesbar
  - Ohne DTD: Keine Dokumentvalidität, keine Semantik für Datenaustausch
  - Mit DTD: Geringere Flexibilität, Dokumente evtl. ungültig bei Änderungen

Datenmodelle/Formate: Zusammenfassung

- "Austauschformate"
  - Entry-based
  - ASN.1
  - XML
- Speichern und Anfragen
  - Relationales Modell
  - Objektorientiertes/Objektrelationales Modell
- Vorteile der Flatfiles nicht unterschätzen
  - Viele Bio-Einrichtungen ohne RDBMS / Informatiker
- I.d.R. mehrere Formate/Datenmodelle in *einem* Bio-Projekt

Spezielle Modellierungsaspekte

- Objektidentifikation
- Versionierung
- Widersprüchliche Daten

Objektidentifikation

- Trivial?
- „Namen" einer Sequenz in Genbank
  - Locus name
  - Accession number
  - GI number
  - NID number
  - Accession.Version
  - Weitere Ids in EMBL / DDBJ
- Existierende Crossreferenzen wechseln dadurch nicht
  - Falsche Verweise
  - Tote Verweise (Dangling Pointer)

Identifikation

- Aspekte
  - Definition von Objekten zur Untersuchung
  - Identifikation von Objekten der realen Welt
  - Identifikation von Datenbankobjekten
- Alle drei in der Bioinformatik nicht trivial
  - „Objekte" verändern sich: EST-Cluster, Proteindomäne, etc.
  - Identifikation identischer Objekte i.d.R. schwierig
    - unzureichenden Daten (Clonenamen)
    - Fehlenden Standards (Protein / Gennamen)
  - Evolution: Splitting, Merging, Deleting, Versioning, ...

Modelle von Identifikatoren

- Semantikfrei
  - Object-Ids
  - „Surrogate" Keys
- Semantikbehaftet
  - „Sprechende Schlüssel"
  - Beispiel: „CYC_BOVIN" = „Protein_Species" (SP)
  - Problem: Können sich ändern
    - Neue Erkenntnisse über Proteinfunktion
    - Neue Erkenntnisse über Vorkommen in Species
    - Heirat
    - ...

Schlüssel in MDB

- Wissen über Objekte wächst ständig
  - Korrekturen – andere Annotation, Sequenz, ...
  - Objektverschmelzung
    - Zwei Loci sind nur einer
    - Zwei Proteinsequenzen sind redundant
  - Objektteilung
    - Ein Locus sind eigentlich zwei
    - Ein Clone sind zwei (Änderung durch Vermehrung)
- Crossreferenzen sehr weit verbreitet
  - In anderen Datenbanken und in Publikationen
- Links sollen erhalten bleiben
- Versionierung notwendig

Beispiel GenBank

- Versuch 1: „Locus name"
  - Sprechend: „HUMHBB" – „Human Betaglobin region"
  - Definitionen verändern sich - als Ids abgelöst
  - ID und Daten können sich ändern
- Versuch 2: Accession-Number
  - Eindeutige, globale ID für jede Submission
  - Keine Versionierung
  - ID bleibt, aber Daten können sich ändern
- Versuch 3: GID (Genbank ID)
  - Eindeutige, interne ID für jede Version einer Submission
  - Versionen von Entries – unterschiedliche gid
  - ID und Daten immer gleich
  - Aber: Zugriff auf „aktuellsten" Entry nicht möglich (Nur über Comment line)

Beispiel GenBank 2

- Versuch 4: NID (Nucleotide ID)
  - Eingeführt als übergreifende GID für EMBL/Genbank/DDBJ
  - Versionen von Entries – unterschiedliche NID
  - ID und Daten immer gleich
  - Aber: Zugriff auf „aktuellsten" Entry nicht möglich (Nur über Comment line)
  - Obsolet sein Versuch 5
- Versuch 5: Accession-No.Version (Seit 1999)
  - Accession-No ist eindeutige, globale ID für jede eingesandte Sequenz
  - Bei Updates eines Eintrags – neue Version, aber keine neue Accession-no
  - Referenzierung der aktuellsten Version: Accession_no
  - Gleiche Accession-No -> gleiches Objekt
  - Gleiche Accession-No.Version -> gleiche Daten

Objektnamen bleiben schwierig

- Semantische Keys (Namen) haben viele Vorteile
  - Menschen verstehen sie
- Standards für Namen setzen sich bislang nicht durch
- Clone – kein Standard
- Loci – kein Standard
- Proteinnamen – kein Standard
- Gennamen – halbherziger Standard (HUGO)
- Enzymnamen – schlechter Standard (Reaktion statt Objekt)
- ...

Objekt Identifikation, Beispiel (Stand 2001)

- TP53 (neu), P53 (alt)
- SIRT1 (neu), SIR2L1 (alt)

HUGO name	GDB	GenAtlas	OMIM	GeneCards	LocusLink
TP53	1	33	52	22	13
P53	1	17	188	69	63
SIRT1	1	0	5	1	2
SIR2L1	0	0	1	1	2

Versionierung

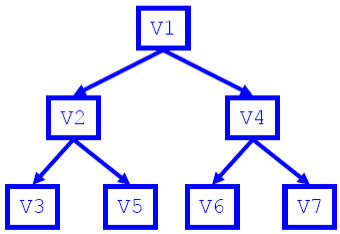
- Das molekularbiologische Wissen wächst und verändert sich ständig
  - Ensembl: ca. 40% Änderungen pro Release
  - Swiss-Prot: Ca. 30% der Einträge ändern sich pro Release
- Analysen benutzen bestimmte Version (aktuellste)
- Nachvollziehbarkeit von Analysen nicht gewährleistet
  - wenn Daten eines bestimmten Release nicht widerherstellbar sind
- Versionierung von Daten essentiell
  - Version der Datenbank (Release)
  - Versionen von Objekten / Attributen
- Verschiedene Aspekte
  - Speichern von verschiedenen Versionen
  - Zugriff auf (alte) Versionen
  - Referenzieren auf Versionen
  - Vergleich von Versionen (Deltas erstellen)

Versionierungsmodelle

Linear

- Zeitliche Reihenfolge fest
- Standardverfahren: Release / Version

Hierarchisch



- Zeitliche Reihenfolge halbgeordnet
- Verfahren von CVS
- Deutlich komplexer

Weitere Kriterien

- Granularität
  - Datenbank
  - Tabelle
  - Objekt (Tuple)
  - Attribut
- Repräsentation von Änderungen
  - Komplette Versionen: Speicherverbrauch
  - Deltas: Berechnung von alten Versionen teuer
- Repräsentation
  - Versionsnummern (Vorgänger / Nachfolger)
  - Timestamps (Nur linear)

Speicherung von Deltas

- Urzustand: A_0, aktuelle Version: A_n
- Schreibweise: Delta($a_x \rightarrow a_y$) ≡ (a_y, a_x)
- Variante 1: $(a_n,a_{n-1}),(a_{n-1},a_{n-2}),...a_0$
  - Aktuelle Version in n Schritten berechenbar
  - Kompakte Deltas
- Variante 2: $(a_n,a_0),(a_{n-1},a_0),...a_0$
  - Aktuelle Version in 2 Schritten berechenbar
  - Zunehmend große Deltas
  - a_0 muss zugreifbar bleiben (Archivierung)
- Variante 3: $a_n,(a_{n-1},a_n),(a_{n-2},a_{n-1}),...(a_0,a_1)$
  - Aktuelle Version immer vorhanden
  - Alte Versionen schwer zugreifbar
  - Kompakte Deltas

Speicherung von Deltas 2

- Variante 4: $a_n,(a_{n-1},a_n),(a_{n-2},a_n),\ldots,(a_0,a_n)$
  - Aktuelle Version sofort zugreifbar
  - Erzeugen einer neuen Version: Neuberechnung aller Deltas
  - Zunehmend große Deltas
- Variante 5: $a_n,a_{n-1},a_{n-2},\ldots,a_1,a_0$
  - Keine Speicherung von Deltas
  - Höchster Speicherplatzverbrauch
  - Alle Versionen im direkten Zugriff

Existierende Versionierung in MDB

- Linear
- Granularität
  - Datenbank Releases
  - Versionierung von Einträgen (Entry-Based Modell)
- Repräsentation
  - Komplette Versionen, keine Deltas (Variante 5)
  - Versionsnummern (Genbank)
  - Änderungsdatum (Swiss-Prot)
- Anmerkung: Wenige MDB „löschen" Objekte
  - Externe Referenzen vorhanden
  - Stattdessen (Ids bleiben gültig)
    - Merging von Objekten
    - Inaktivierung

Senario

- Einspielen von Datenbankreleases in ein versioniertes Schema
- Applikation vergleicht aktuelles Release R_x mit Vorgängerrelease R_{x-1}
  - Objekt K in R_x und nicht in R_{x-1}: `INSERT k`
  - Objekt K in R_x und in R_{x-1}
    - Unterschied: `UPDATE K`
    - Kein Unterschied: Nichts tun
  - Objekt K nicht in R_x aber in R_{x-1}: `DELETE k`
- Annahme: Keine Schlüsselwiederverwertung
  - Schlüssel K wird niemals für verschiedene Objekten benutzt
    - Kann z.B. durch Sequenz sichergestellt werden
  - Objekte können nicht wieder belebt werden

Versionen im relationalen Modell

- Aufgabe
  - RDBMS soll Versionen verwalten
  - Lineare, tuple-basierte Versionierung
- Anforderungen: Zugriff muss möglich sein auf
  - Aktuelle Version
  - Zustand der Datenbank zu beliebigem Zeitpunkt d_0
- 2 Varianten
  - Single Table
  - Schattentabellen

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=true, D=SYSDATE)
  - Ja
    - Letzte Version von K in T finden (V_x)
    - INSERT K INTO T (V=V_x+1, A=true, 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=false, 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=true, D=SYSDATE)
      - Nein
        - » Nichts tun – Fehler?

Variante 1: SELECT

- SELECT aktuelle Version von k_0

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

- SELECT alle Tupel zum Zeitpunkt d_0

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

Was könnte man sparen ?

Variante 1: Bewertung

- Varianten
  - Versionsnummer weglassen (Datum reicht)
  - Markierung der aktuellen Version hilfreich
    - Nur in aktuellster Version gilt A=T
    - INSERT/UPDATE erfordert 1INSERT + 1UPDATE
    - Zugriff auf aktuelle Version schneller
- Bewertung
  - INSERT / DELETE / UPDATE erfordern Trigger
  - Verlangsamung bei SELECTs
  - Verlangsamung durch wachsende Tabelle
  - Syntaxkomplexität durch Views abfangen

Variante 2: Schattentabellen

- Pro Tabelle T anlegen einer Tabelle T^s
  - Zusätzliche Attribute
    - Versionsnummer V
    - VALIDUNTIL D
  - Schlüssel in T^S: 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 aktuelle Version von k_0

```
SELECT *
FROM t
WHERE k=k0
```

- SELECT alle Tupel k zum Zeitpunkt d_0
  - Kompliziert ....
  - Erfordert Zugriff auf T und T^S
  - Vorsicht: Datum in T^S gibt Ende der Gültigkeit an

Fallunterscheidung

	Gelöscht vor d_0	Gelöscht nach d_0	Bisher nicht gelöscht
Letzte Änderung vor d_0	K nicht in T; Kein Eintrag in T^S mit $d>d_0$	K nicht in T; Kleinsten Eintrag in T^S mit $d>d_0$ nehmen (es gibt nur einen) 1	K in T nehmen; Bedingung: kein Eintrag in T^S mit $d>d_0$ 2
Letzte Änderung nach d_0	Fall unmöglich	K nicht in T; Kleinsten Eintrag in T^S mit $d>d_0$ nehmen (es gibt mehrere) 3	K in T vorhanden (ignorieren); Kleinsten Eintrag in T^S mit $d>d_0$ nehmen (es gibt mehre 4

In SQL

```
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
       t_s.d =
         (SELECT MIN(d)
          FROM t_s t2
          WHERE t_s.k=t2.k AND
                t_s.d > d0)
```

Fall 2

Fälle 1,3,4

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
- Seltenere Änderungen, vor allem Zugriff auf aktuellste Version – Variante 2
- Variante 2 eher für MDB geeignet
- Außerdem zu klären
  - Referenzen / Fremdschlüssel auf versionierte Objekte
    - Entry-based Daten haben keine Referenzen
  - Identifikation von Änderungen (Delta Berechnung)

Widersprüchliche Daten

- Experimentergebnisse können (und werden) sich "widersprechen"
  - "Widersprüche" sind eine Frage des zugrundeliegenden, angenommenen kausalen Modells
  - Auflösung der Widersprüche i.d.R. nicht möglich; zusätzliches Experiment liefert oft nur weiteres Ergebnis
  - "Widersprüche" lösen sich bei Modellüberarbeitung oft auf
- Rohdaten versus Ergebnisdaten
- Metadaten wichtig
  - Datenquelle
  - Motivation
  - Zeitpunkt
  - Abschätzung der Datenqualität
  - Vergleichbarkeit der Ergebnisse
- Bedeutung der Versionierung

Zusammenfassung

- Datenmodelle
  - Flat-file basierte Datenmodelle traditionell weit verbreitet
  - XML-basierte Modelle von zunehmender Bedeutung
  - langfristige Rolle relationaler/objektrelationaler Modelle noch unklar
- Für Bio-DB wichtige Modellierungsaspekte
  - Objektidentifikation
  - Versionierung
  - widersprüchliche Daten