

Molekularbiologische Datenbanken

Spezielle Modellierungsaspekte



Ulf Leser

Wissensmanagement in der
Bioinformatik



Entry-Based

- Datenbank = Menge ähnlich strukturierter **Entries**
- Entry: Menge von **Feldern** (Attribute, Lines),
 - Identifikation durch 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
- Menschenlesbar, leicht zu ändern, sehr flexibel
- Schwer zu parsen, schlechte Skalierbarkeit

Relationales Modell

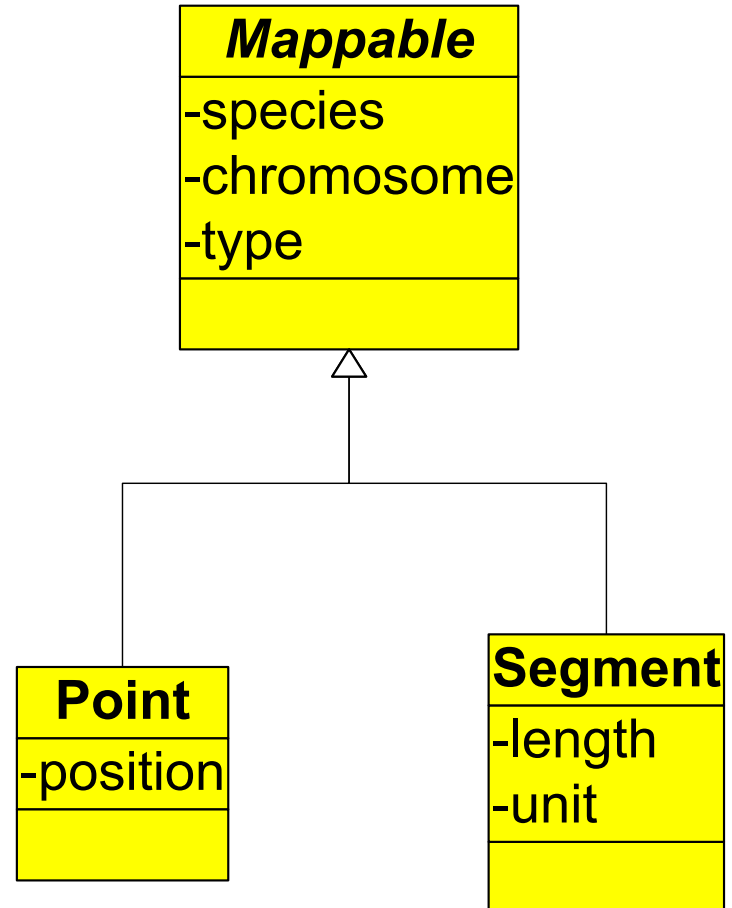
- Modellierung: ER, UML
- Speicher und Anfrageformat (nicht Repräsentation)
- Vorteile
 - Strukturierte Anfragen
 - Sehr weit verbreitet, robust, Industriestandard
 - Skalierbarkeit und Optimierbarkeit
- Nachteile
 - SQL schwierig zu lernen
 - Volltextsuche nicht direkt möglich
 - Erfordert speziell zu entwickelnde Editoren, Suchmasken, ...
 - Erfordert Administration, Backup, etc.

UML

- Unified Modelling Language
 - Klassendiagramme
 - Vorteile
 - Reichhaltige Datenmodell, graphische Notation
 - Automatische Erzeugung von Persistenzschicht mgl
 - Nachteile
 - Dualität OO – RDBMS nicht trivial (Impedance Mismatch)
 - Roundtrip – Problem
 - Queries adressieren rel. Schema, nicht UML Modell
- UML ist für Software, nicht für Datenbanken gemacht

Objektrelationales Mapping

- Vier Varianten
 - **Union**: 1 Tabelle
 - **Vertikale Zerlegung**: 1 schmale Tabelle pro Klasse
 - **Horizontale Zerlegung**: 1 breite Tabelle pro Klasse
 - Volle **Materialisierung**



MapHierarchy					
class_id	species	chromo	length	unit	pos
1	h.s.	X			
1	m.m.	21			
2	h.s.	12			12q2
3	m.m.	X	1.5	cM	

Point	
o_id	pos
3	12q2
4	12q3

Mappable		
o_id	species	type
1	h.s.	STS
2	m.m.	EST
3	h.s.	STS
4	h.s.	STS
5	m.m.	STS
6	m.m.	STS

Segment		
o_id	length	unit
5	200	KB
6	1.5	cM

Mappable	
species	chromo
h.s.	X
m.m.	21

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

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

Point		
species	chromo	pos
h.s.	12	12q2
h.s.	13	12q3

Point		
species	chromo	pos
h.s.	12	12q2
h.s.	13	12q3

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

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

XML - Verwendung

- XML ist ein Austauschformat
- Keine Ablösung von Flatfiles als von Menschen lesbare Repräsentation
- Keine Ablösung von RDBMS als primäres Speichermedium
- Benutzung zur Verteilung computerlesbarer Daten

Inhalt dieser Vorlesung

- Spezielle Modellierungsaspekte
 - Objektidentifikation
 - Versionierung
 - Widersprüchliche Daten
 - Controlled Vocabularies / Ontologien

1. 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 (Clonennamen)
 - 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)
- ...

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 wiederherstellbar 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)

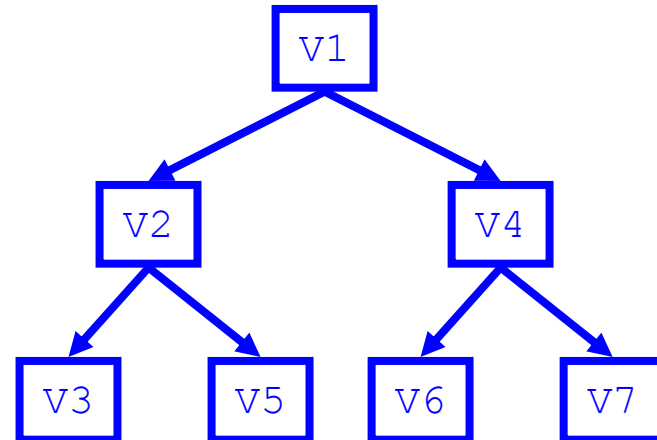
Versionierungsmodelle

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

Speicherung von Deltas – 2

- Variante 4: $a_n, (a_{n-1}, a_n), (a_{n-2}, a_n), \dots, (a_0, a_n)$
 - Aktuelle Version sofort zugreifbar
 - Erzeugen einer neuen Version: Neuberechnung aller Deltas
 - Zunehmend große Deltas
- Variante 5: $a_n, a_{n-1}, a_{n-2}, \dots, a_1, a_0$
 - 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

Szenario

- 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)
```

Fall 2

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)
```

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

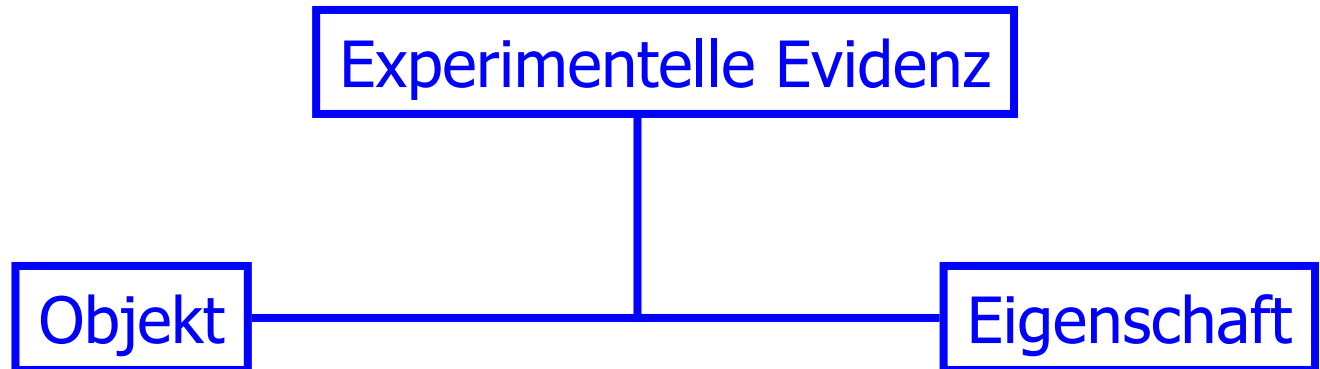
- Häufige Änderungen, eher wenig Lesezugriffe - Variante 1
- Seltenerer Änderungen, vor allem Zugriff auf aktuellste Version – Variante 2
- Variante 2 eher für MDB geeignet
- Außerdem zu klären
 - Referenzen / Fremdschlüssel auf versionierte Objekte
 - Entry-based Daten haben keine Referenzen
 - Identifikation von Änderungen (Delta Berechnung)

3. Widersprüchliche Daten

- Sind sehr häufig
 - Experimentelle Datenbanken: unterschiedliche Ergebnisse
 - Integrierte Datenbanken: unterschiedliche Ansichten
- Repräsentation von Widersprüchen
 - Interpretation aller Fakten als „Observations“
 - Jede Observation hat Evidenz und Quelle
- Observation versus Version
 - Versionen sind geordnet
 - Neue Versionen „besser“ als alte

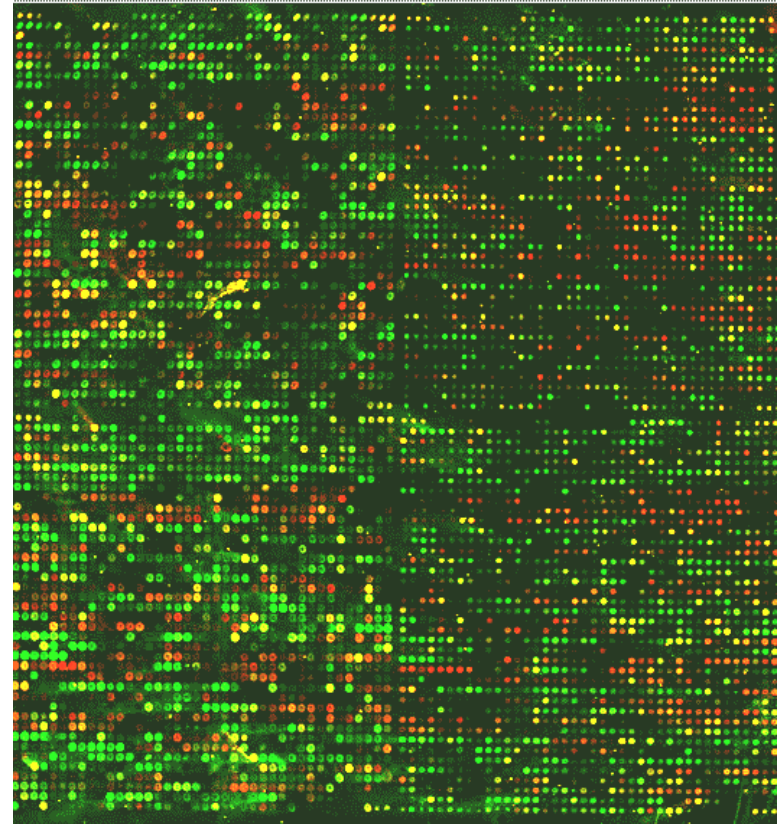
Experimentelle Datenbanken

- Experimentergebnisse können (und werden) sich widersprechen
- Rohdaten versus Ergebnisdaten
- Metadaten wichtig
 - Abschätzung der Datenqualität
 - Vergleichbarkeit der Ergebnisse
- Muster

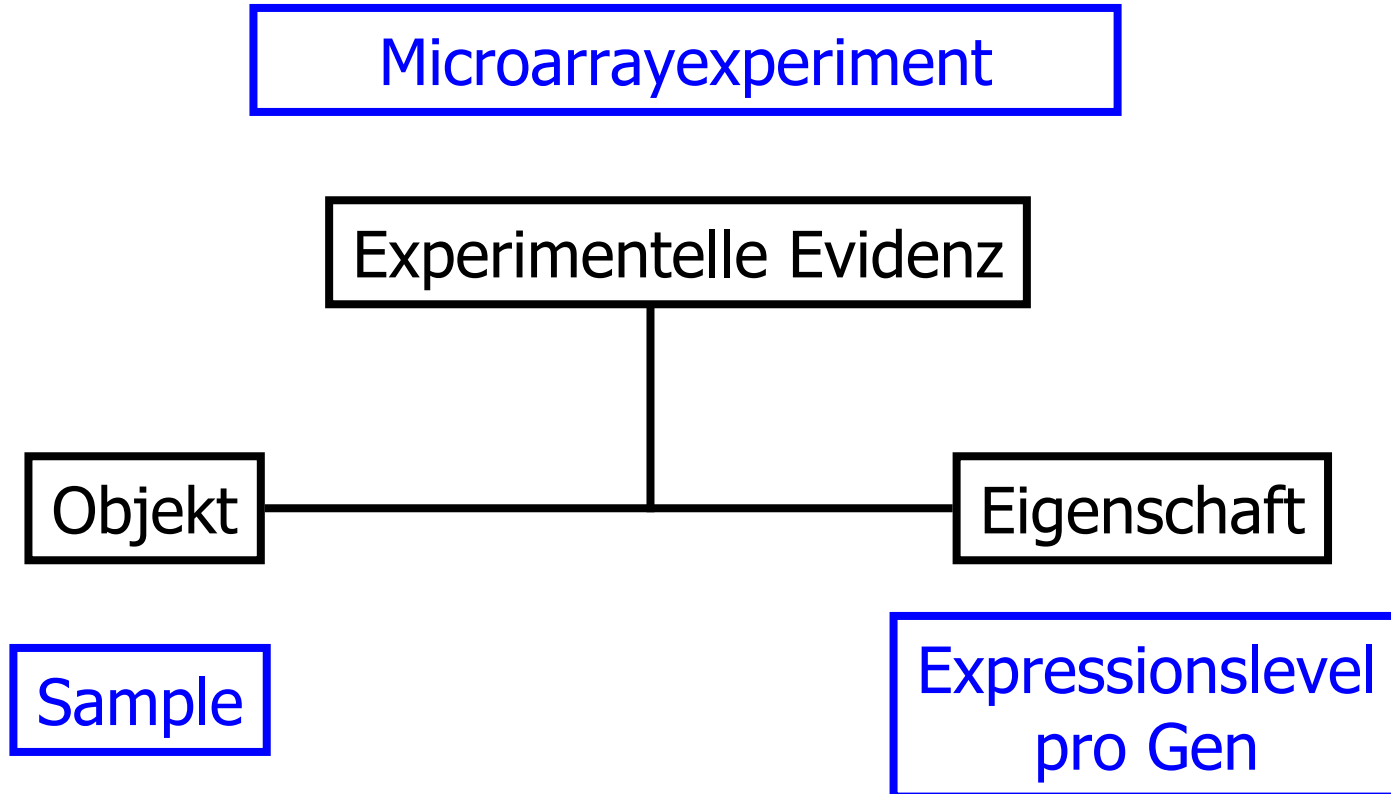


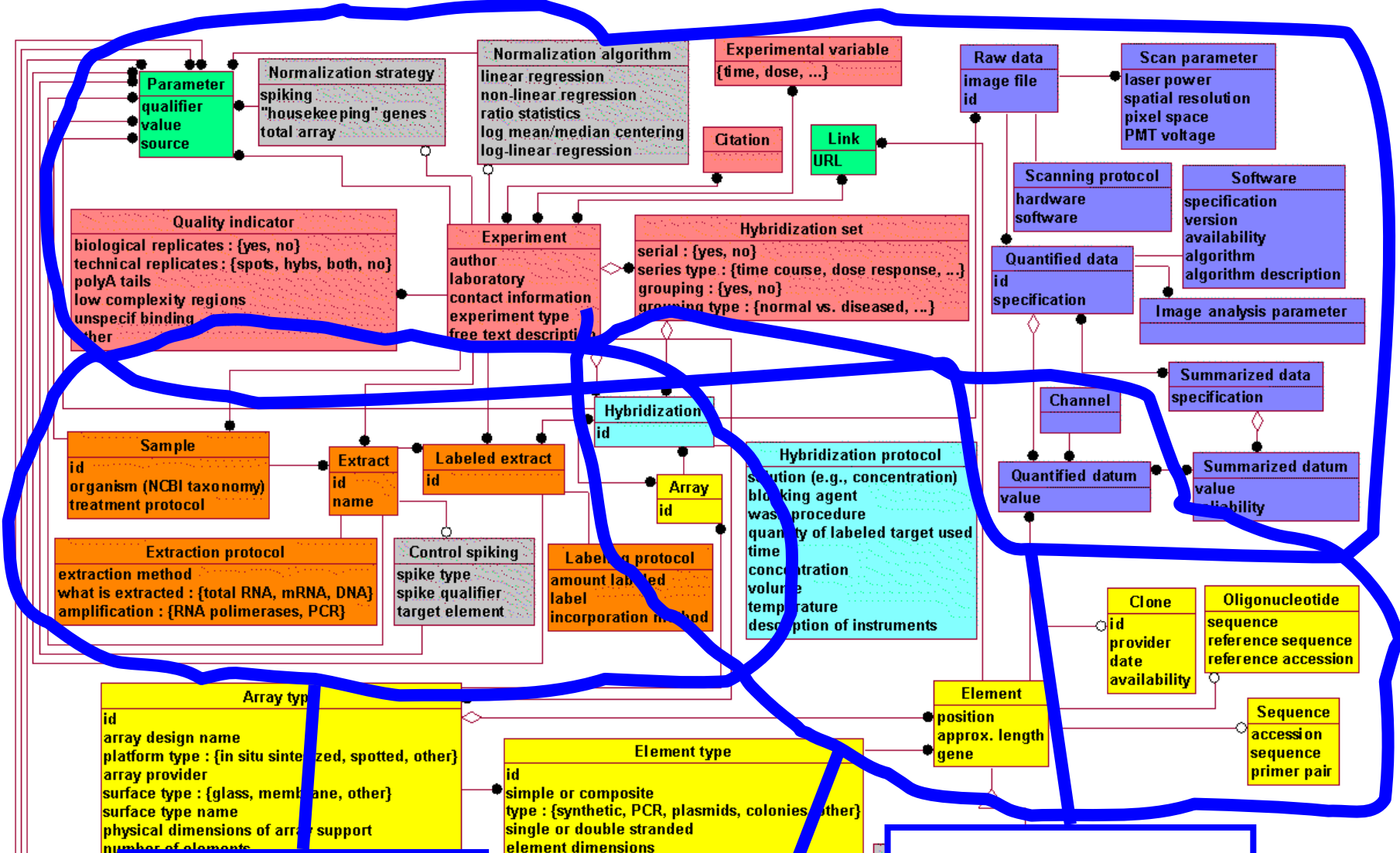
Microarraydaten

- Jede Zelle repräsentiert ein **Gen**
- Farbe: **Expressionsniveau** in Zelle Z zum Zeitpunkt T
- Ziel: Unterschiede im Expressionsniveau zwischen Z_{gesund} und Z_{krank}



Experimente





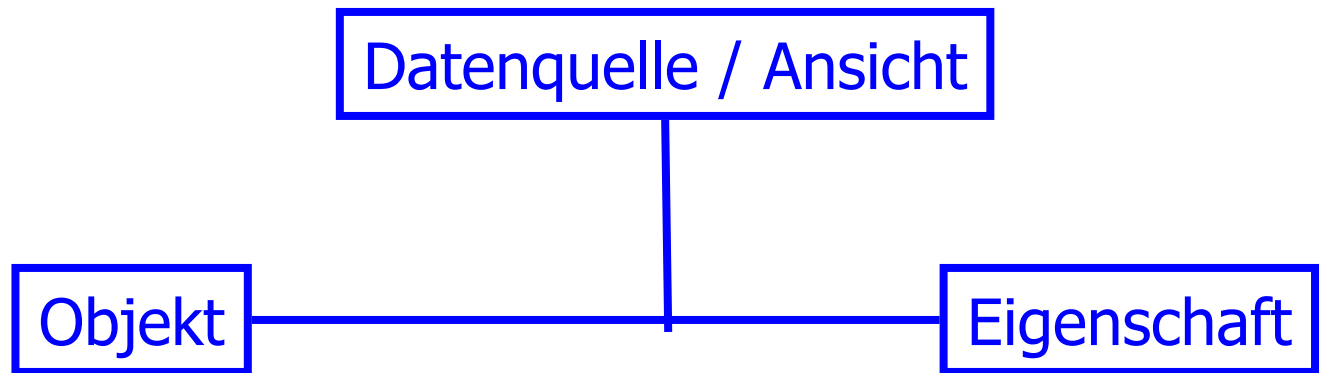
Objekt

Eigenschaft
(Expressionslevel)

Referenz

Integrierte Daten

- Auflösung der Widersprüche i.d.R. nicht möglich
 - Messen liefert nur weiteres Ergebnis
 - „Wahrheit“ nicht vorhanden
- Metadaten wichtig
 - Datenquelle
 - Grundlage der Ansicht
 - Zeitpunkt
 - ...
- Modell



Modellierung

- Sehr ähnlich zu experimentellen Daten
- Praktisch jede Eigenschaft eines biologischen Objektes kann widersprüchlich beschrieben werden
 - Clonelänge, -typ, -herkunft (Gewebe)
 - Sequenzabschnitte, -basen, -motive
 - ...
- Extremfall: **Keine einwertigen Attribute**

4. Mind your words

- Anfrage: „Suche alle Sequenzen des humanen X Chromosoms“
- 1996 nicht beantwortbar
- Grund
 - Source: Homo Sapiens
 - Source: Human
 - Source: H.S.
 - Source: hs
 - Source: french girl with Huntington's Disease
 - Source: ...

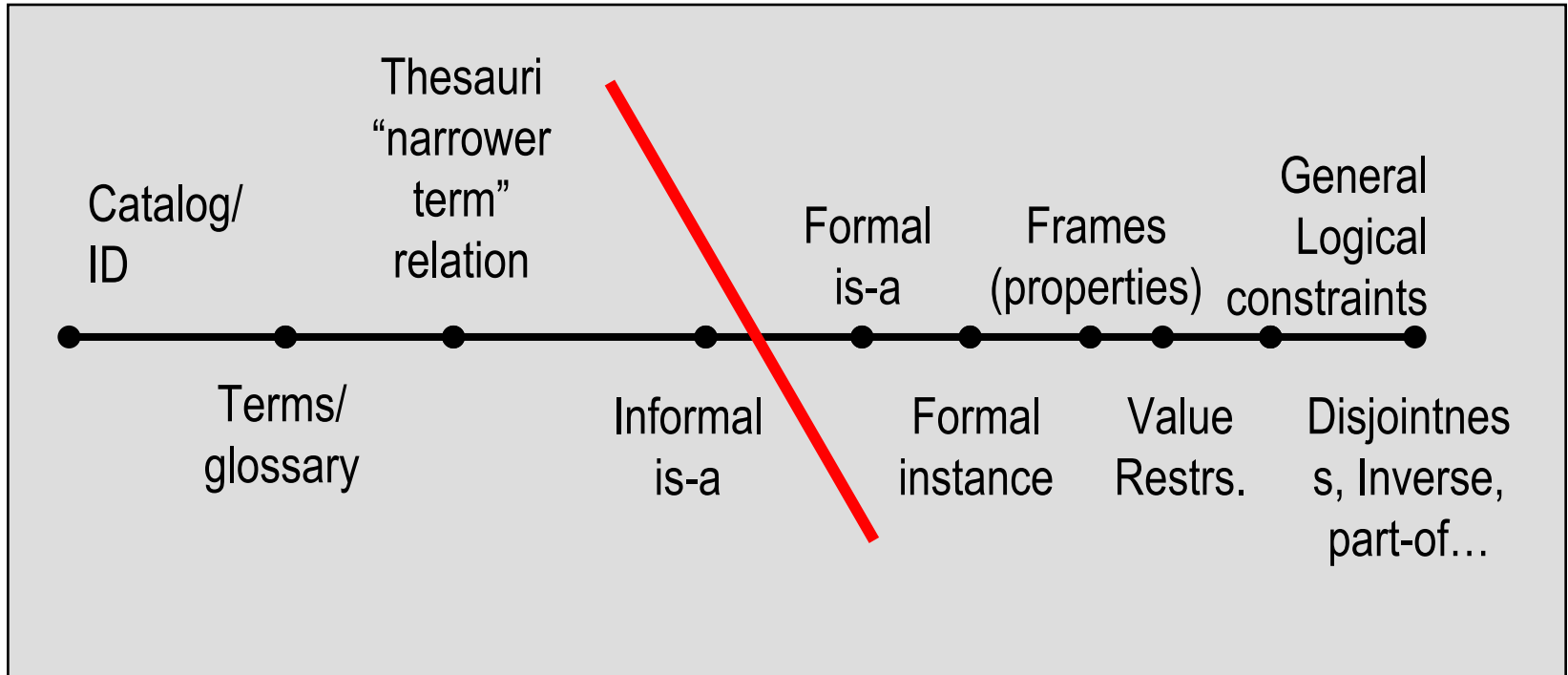
Controlled Vocabularies

- Zweck
 - Präzise Suchergebnisse
 - Validierung von Eingabemasken
 - Vergleichbarkeit von Daten: Proteinfunktion in verschiedenen Spezies, enzymatische Eigenschaften, ...
- Viel Aufmerksamkeit in den letzten Jahren (endlich)
 - Keywordlisten
 - Taxonomy Projekt
 - GeneOntology
- Vokabulare, Thesauri, Schlüsselwörter, ...
- Ontologien

Ontologies

- ... specification of a conceptualization ... [Gruber, 1995]
- Aim
 - Write down everything a group of people knows about / agrees upon a specific domain such that the result can be used for a specific application
- Properties
 - Domain dependent
 - Application dependent
 - For communication, data exchange, reuse
- Roots
 - Knowledge bases & exchange: KIF, KQMS, etc.
 - Knowledge representation: KL-ONE, Frames languages
 - Classifications and thesauri: libraries, info. retrieval

Ontology Formalisation



Source: Robert Stevens, From Building and Using Ontologies, U. of Manchester

Example from SwissProt

Entry name	1433_ARATH
Accession number	P42644
Created	Rel. 32, 1-NOV-1995
Sequence update	Rel. 32, 1-NOV-1995
Annotation update	Rel. 41, 15-JUN-2002
Description and origin of the Protein	
Description	14-3-3-like protein GRF3 (General regulatory factor 3) (14-3-3-like protein RCI1).
Gene name(s)	GRF3 OR RCI1 OR AT5G38480 OR MFL10.21.
Organism source	Arabidopsis thaliana (Mouse-ear cress).
Taxonomy	Eudicotyledons; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta; Spermatophyta; Magnoliophyta; eudicotyledons; Core eudicots; Rosidae; eurosids II; Brassicales; Brassicaceae; Arabidopsis.
NCBI TaxID	3702

Comments	
FUNCTION	IS ASSOCIATED WITH A DNA BINDING COMPLEX THAT BINDS TO THE G BOX, A WELL-CHARACTERIZED CIS-ACTING DNA REGULATORY ELEMENT FOUND IN PLANTS GENES.
INDUCTION	BY COLD.
SIMILARITY	BELONGS TO THE 14-3-3 FAMILY.

50	Sequence	255 BP;	
	MSTREENVYM AKLAEQARY EEMVEFMERK AKTVDVRELS VEERNLLSVA YKNVICARRA		60
	SWRIISSIEQ KEESKGMEDH VAIKDYRGE IESLSKICD GILNVLEAHL IPSASPASK		120
	VFYLRKMGDY HRYLAEFKAG AERKEAAEST LVAYKSADI ATAEPLAPHP IRLGLALNFS		180
	VFYYEILNSP DRACSLAKQA FDDAIAELDT LGESYKST LIMQLLRDNL TLWTSDMTDE		240
	AGDEIKKASK PDGAE		255

GeneOntology

- Founded and maintained by biologists
- Goals
 - Compile structured vocabularies describing **certain** aspects of molecular biology
 - Use these vocabularies to annotate database objects (especially genes and gene products)
 - Develop tools for editing and using the GeneOntology
- Three vocabularies
 - Biological processes (6742 terms)
 - Molecular function (5299 terms)
 - Intra-cellular location (1125 terms)

Example from InterPro

Reset View InterProEntry

This entry is from: [INTERPRO](#)
Save Link
Printer Friendly

Glucose-methanol-choline oxidoreductase

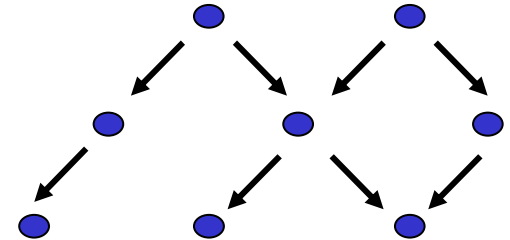
Accession	IPR000172; (GMC_oxred) matches 174 proteins
FullName	Glucose-methanol-choline oxidoreductase
Type	Family
Signatures	PROSITE: PS00623 GMC_OXRED_1 PROSITE: PS00624 GMC_OXRED_2 PFAM: PF00732 GMC_oxred
Biological Process	electron transport (GO:0006118)
Molecular Function	electron transfer flavoprotein (GO:0008246)
Abstract	The glucose-methanol-choline (GMC) oxidoreductase oxidoreductases are FAD flavoproteins oxidoreductases [1, 5]. These enzymes include a variety of proteins; choline dehydrogenase (CHD), methanol oxidase (MOX) and cellobiose dehydrogenase [EC:1.1.5.1] [6] which share a number of regions of sequence similarities. One of these regions, located in the N-terminal section, corresponds to the FAD ADP- binding domain. The function of the other conserved domains is not yet known.
Examples	<ul style="list-style-type: none">• P22637 Cholesterol oxidase (CHOD) () from Brevibacterium sterolicum and Streptomyces strain SA-COO.• P13006 Glucose oxidase () (GOX) from Aspergillus niger.• O50048 (R)-mandelonitrile lyase () (hydroxynitrile lyase) from plants [PUB00004524].• P54223 Choline dehydrogenase () (CHD) from bacteria.• P18173 Glucose dehydrogenase (GLD) () from Drosophila.

Document: Done (2.794 secs)

- Replace (enhance) annotation with GO terms
 - Standardized term
 - Standardized definition
 - Standardized relationships to other terms

Structure of GeneOntology

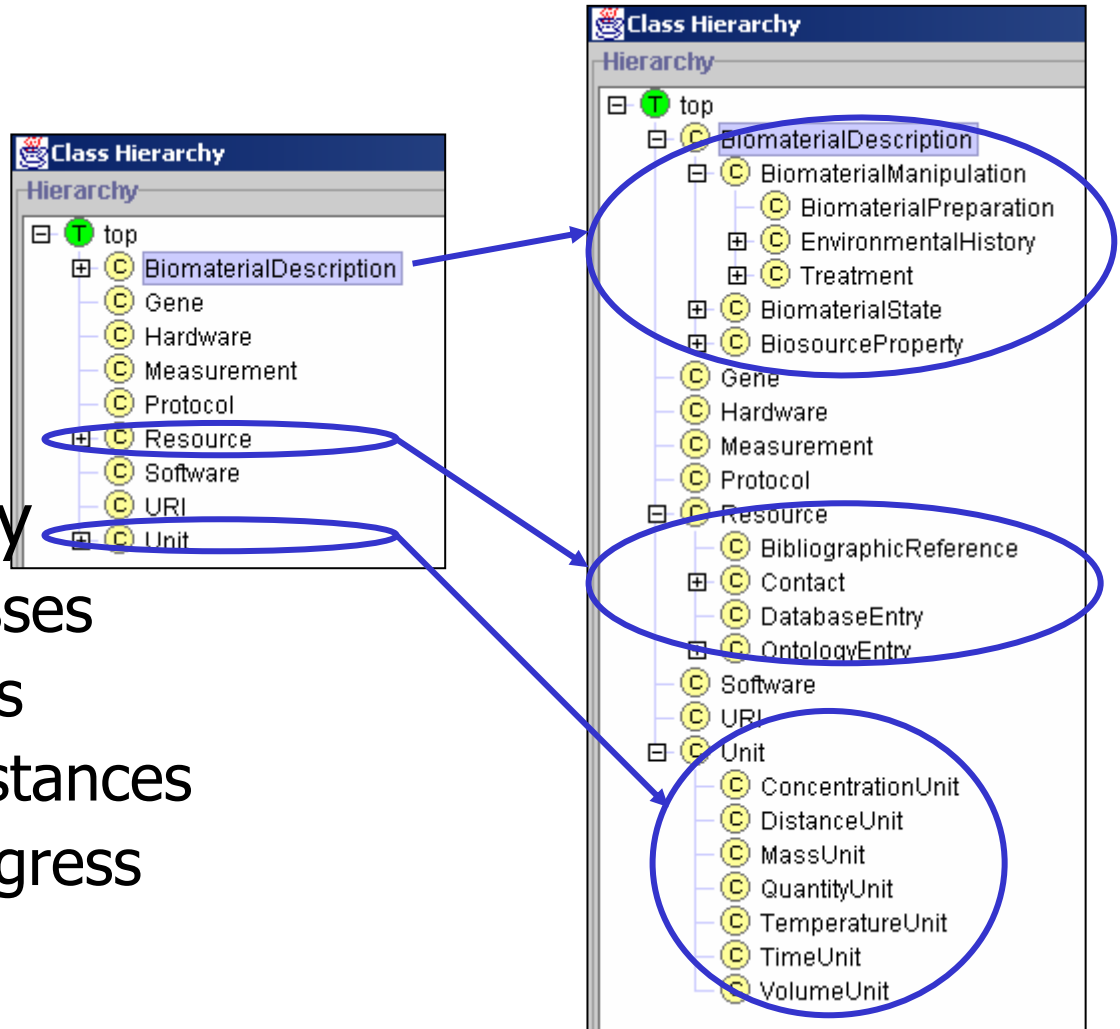
- A GO ontology is
 - A collection of terms
 - Each term has a (free text) description
 - Each term has a unique ID (GO:XXXX)
 - Terms may have synonyms and DB-Xrefs
 - IS-A relationships
 - PART-OF relationships
 - IS-A and PART-OF relationships form (each) an acyclic graph
 - True Path rule: „Every path from a node back to the root must be biologically accurate“



MGED Ontology

Size of ontology

- app. 80 classes
- app. 70 slots
- app. 100 instances
- Work in progress



UMLS parts

<p>Entity</p> <ul style="list-style-type: none"> Physical Object Organism <ul style="list-style-type: none"> Plant <ul style="list-style-type: none"> Alga Fungus Virus <u>Rickettsia or Chlamydia</u> Bacterium <u>Archaeon</u> Animal <ul style="list-style-type: none"> Invertebrate Vertebrate <ul style="list-style-type: none"> Amphibian Bird Fish Reptile Mammal <ul style="list-style-type: none"> Human Anatomical Structure <ul style="list-style-type: none"> Embryonic Structure Anatomical Abnormality <ul style="list-style-type: none"> Congenital Abnormality Acquired Abnormality Fully Formed Anatomical Structure <ul style="list-style-type: none"> Body Part, Organ, or Organ Component Tissue Cell <ul style="list-style-type: none"> Cell Component Gene or Genome Manufactured Object <ul style="list-style-type: none"> Medical Device Research Device Clinical Drug 	<ul style="list-style-type: none"> isa associated with physically related to <ul style="list-style-type: none"> part of consists of contains connected to interconnects branch of tributary of ingredient of spatially related to <ul style="list-style-type: none"> location of adjacent to surrounds traverses functionally related to <ul style="list-style-type: none"> affects <ul style="list-style-type: none"> manages treats disrupts complicates interacts with prevents brings about <ul style="list-style-type: none"> produces causes 	<ul style="list-style-type: none"> [associated with] (continued) [functionally related to] (continued) performs <ul style="list-style-type: none"> carries out exhibits practices occurs in <ul style="list-style-type: none"> process of uses <ul style="list-style-type: none"> manifestation of indicates result of temporally related to <ul style="list-style-type: none"> co-occurs with precedes conceptually related to <ul style="list-style-type: none"> evaluation of degree of analyzes <ul style="list-style-type: none"> assesses effect of measurement of measures diagnoses property of derivative of developmental form of method of conceptual part of issue in
--	--	--



ing
ce