

Vorlesungsplan

1. Übersicht
2. Datenmodelle
 - Textdateien, Entry-Modell
 - Relationale DB
 - XML
3. Genom-DB
4. Genexpressions-DB
5. Protein-DB
6. Pathway-DB
7. Publikations-DB
1. OWL & Ontologien
- 8. Zugriff und Anfragesysteme**
9. Datenintegration von Bio-DB
10. Webservices

Zugriffsmethoden in Bio-Datenbanken

n Navigation

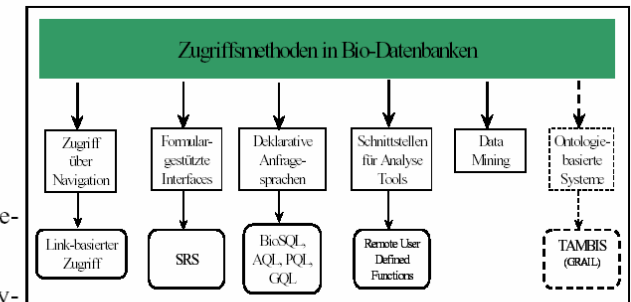
n Stichwortsuche

- Formulargestützte Interfaces
- SRS

n Deklarative Anfragesprachen

n Schnittstellen für Analyse-Tools

n Data Mining



Navigation

n Ansatz

- Browsen in den Datenbeständen; Zugriff auf benötigtes (Zusatz)-Wissen über html-Links

n Vorteile

- Einfach zu realisieren
- Für Standardfälle effizient

n Problematik

- Wenig flexibel, kein Muster-basierter Zugriff
- Verlinkung unterliegt der "Willkür" der DB-Anbieter
- "Lost in Hyperspace"-Phänomen
- Referentielle Integrität schwierig zu wahren

n Wesentliche größere Flexibilität erst durch Bio-Ontologien

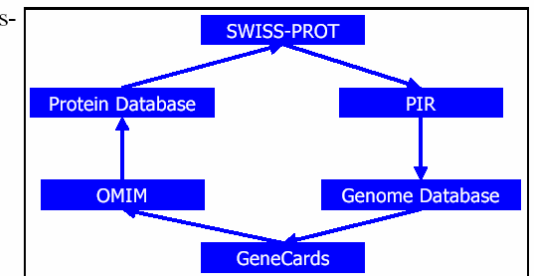
- Ontologie: Explizite begriffliche Formalisierung eines Anwendungsbereiches (Fachsprache)
- Dazu mehr in Kapitel 8 (Integration von Bio-Datenbanken)

Navigation: Beispiel

n Stichwort: Duchenne Muskeldystrophie

n Startpunkt:

- Swiss-Prot (EBI) mit Stichwort "Duchenne" (DMD_HUMAN; menschliche Duchenne Muskeldystrophie)



Swiss-Prot: Eingabe

The screenshot shows the Swiss-Prot search interface. On the left, there are search options including selecting databases, entering search terms, and browsing through entries. On the right, a list of available databases is shown, categorized into Literature, Nucleotide sequence, Protein sequence, and Nucleotide related databases. The 'Protein sequence' category is expanded, showing databases like SWALL (SPTR), Swiss-Prot, SpTReMBL, TrEMBL (Updates), IPI, RemTReMBL, PIR, REFSEQP, PATENT_PRT, JPO_PRT, USPTO Proteins, and MHCBI.

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 543

Swiss-Prot: Ausgabe (Auszug)

- n Kurze Beschreibung (mit Lit.-Referenzen etc.)
- n Link u.a. zu Datenbank PIR (Protein Information Resource): A27605

The screenshot shows the Swiss-Prot entry for DMD_HUMAN. The 'General information' section includes the entry name, accession number (P11532, Q14170), creation date, sequence update, and description. The 'Description and origin of the protein' section provides details about the protein's function and its location. A list of related sequences is shown on the right, with the entry A27605 highlighted in blue.

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 544

PIR-Eintrag

- n Link u.a. zu Datenbank GDB (Genome Database): GDB:119850

The screenshot shows the PIR entry for DMD. The 'General Information' section includes the ID, accession number, date, description, species, sequence length, keywords, and comments. A diagram of the protein structure is shown on the right, with the entry A27605 highlighted in blue.

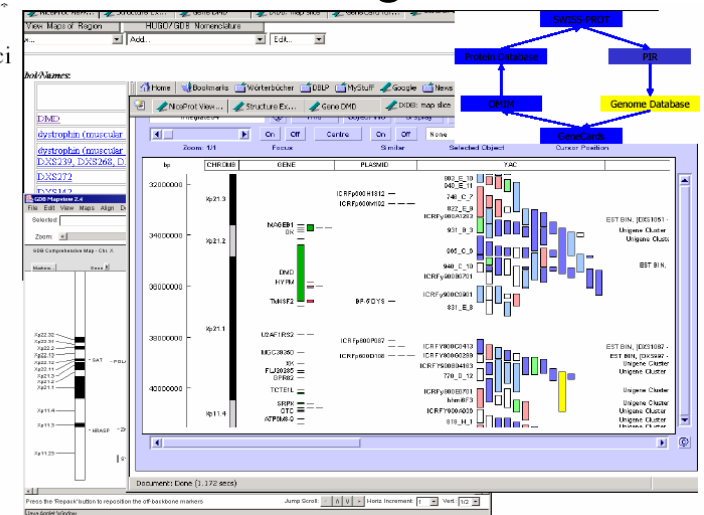
Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 545

GDB-Eintrag

- n Genome Database*
- n Aliase, Clone, Loc
- n Karten
- n Link to GeneCard DMD



* <http://gdbwww.dkfz-heidelberg.de/>
Sommersemester 2005/06

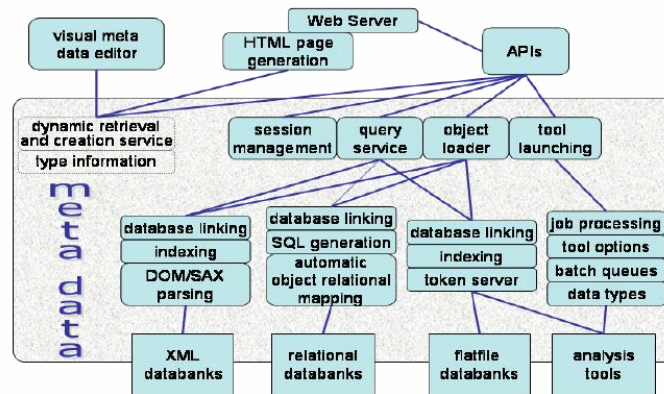
Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 546

SRS (Sequence Retrieval System)

- n** Ursprünglich als Zugriffstool für Sequenzdatenbank EMBL entwickelt

- Mehr als 400 wissenschaftliche Datenbanken ansprechbar
- Einheitliche graphische Nutzerschnittstelle
- Formularbasiert
- Fokus auf Flatfile-Datenbanken, Weiterentwicklung zur Unterstützung relationaler Datenquellen
- Übernahme und kommerzieller Vertrieb



Source: LionBioScience

Sommersemester 2005/06

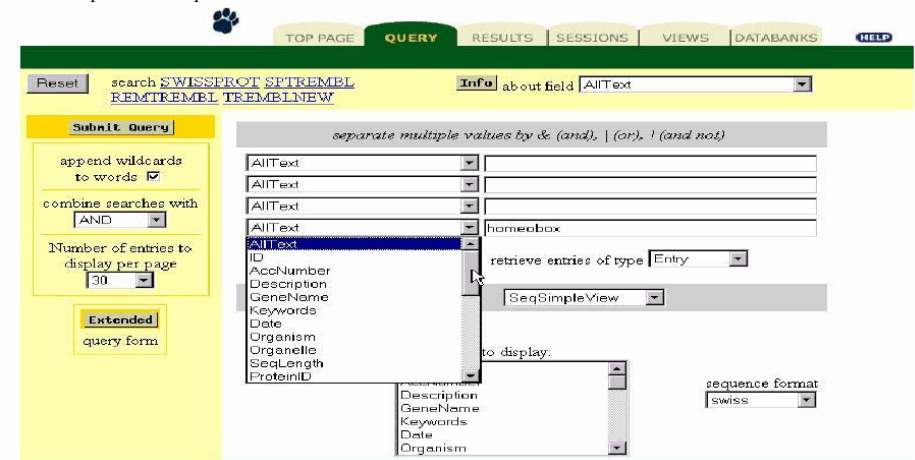
Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 551

SRS: Konzept (1)

- Ein "Konzept" wird über eine Anzahl von Feldern (Entries, Attributen) beschrieben

- Beispiel: Konzept $\mathcal{E}n$



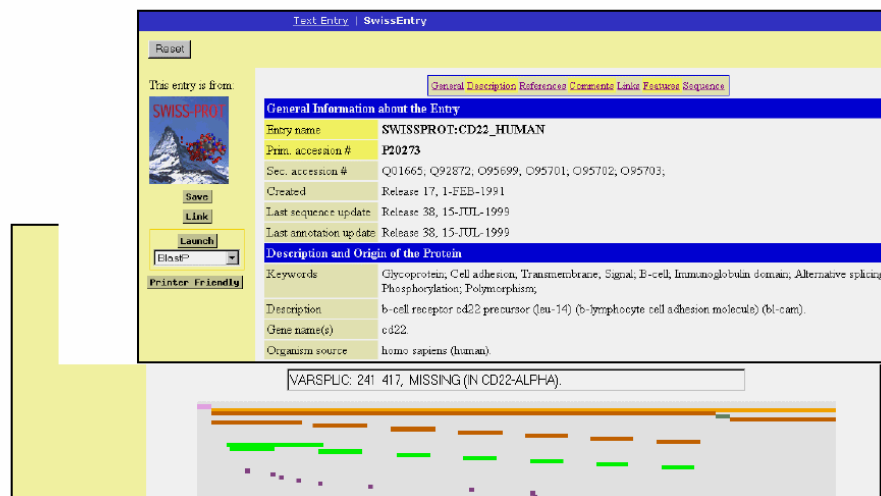
Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 552

SRS: Konzept (2)

- n Ein Konzept kann als html-Seite visualisiert werden



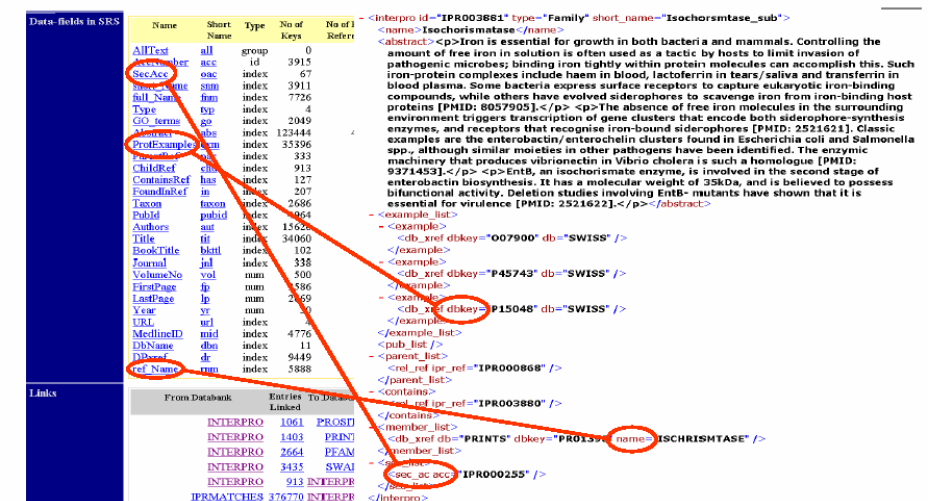
Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 553

SRS: Konzept (3)

- n** Export als SRS-"Objekt" (C++, Java, Perl, CORBA) oder Speicherung als XML



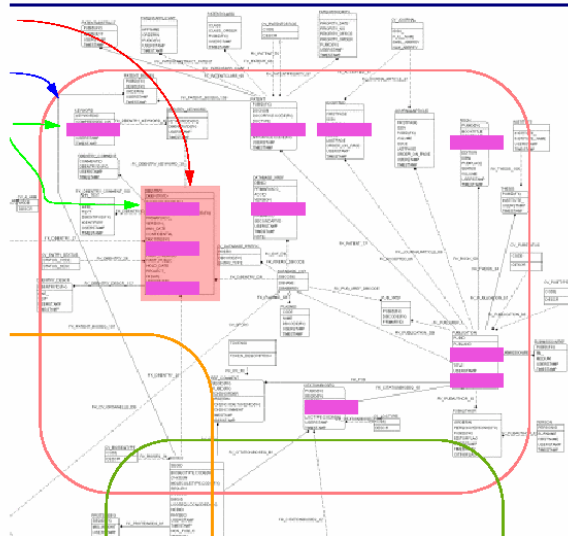
Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 554

SRS: Zugriff auf relationale Datenbanken

- n Auswahl einer *hh*-Tabelle (als "Aufhänger"-Konzept) (→)
- n Angabe der Tabellen die zum Konzept "dazu gehören" (→)
- n Angabe der abfragbaren Attribute (→)
- n Interne Umsetzung via Views/Joins



Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 555

SRS: Datenzugriff (1)

- n Objektsuche
 - Auswahl von Datenquellen
 - Spezifikation der Suchkriterien für abfragbaren Attribute
 - Schnittmenge der Attribute aller ausgewählten Quellen (z.B. ID)
- n Unterstützte Query-Möglichkeiten
 - Textsuche, Bereichsuche für numerische/Datumattribute
 - Regulär Ausdrücke
- n Automatische Übersetzung von SRS-Queries nach SQL zum Zugriff auf relationale Datenbanken
- n Ergebnis als Vereinigung der Suchergebnisse über einzelne Quellen

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 556

SRS - Datenzugriff (2)

- n Querverweis-Suche
 - Für Objekte einer Ergebnismenge oder für eine ganze Quelle
 - Referenzierte Objekte in anderen Datenquellen
- n Automatische Bestimmung der Pfade zwischen Quellen
 - Shortest-Path-Algorithmus
 - Gleiche Semantik der Beziehungen
- n Tool-Integration
 - Anwendung auf Ergebnismengen der Abfragen
 - Anzeige direkt im Web-Browser
 - Große Anzahl von Tools bereits integriert

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 557

SRS Anfrage Sprache

- Anfragen können an vielen Stellen direkt formuliert werden, z.B. Results -> Search using a query expression
- Anfrage-Syntax
 - Stringsuche: [Menge-Mengenattribut:Suchmuster], Menge kann eine Datenbank, DB-Gruppe, Index, Index-Gruppe oder Suchausdruck sein
 - z.B. [pir-des:elestase], oder [swissprot-AllText:duchenne*]
 - Wildcards: [swissprot-aut:sanger,*!coulson,a*]
 - Reguläre Ausdrücke: [swissprot-aut:/mue?ller/]
 - Zahlenbereiche: [swissprot-SeqLength#400:500]
 - mehrere DBs: [{swissprot swissnew sptrembl}-des:kinase], [dbs={swissprot swissnew sptrembl}-des:kinase] &[dbs-org:human]

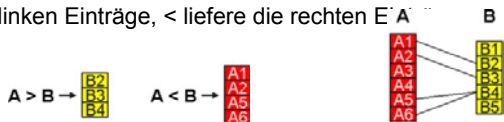
Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 558

SRS Anfrage Sprache

- Einfache Anfragen kombinieren:
 - operand operator operand ...
 - z.B. verlinken: [swissprot-AllText:duchenne*] > pdb
- Operatoren
 - logisch: | oder, & und, ! aber nicht
 - Links:
 - > liefere die linken Einträge, < liefere die rechten Einträge



- Komplexe Verweise: (q = [{swissprot swissnew}-des:kinase])!(q<swissnew)

- Hierarchische Suche
 - >^ liefere Teilbaum-Einträge definiert durch linke Seite
 - >_ liefere Blatt-Einträge des Teilbaums definiert durch linke Seite

SRS Anfrage Sprache

- Mehrfache Links
 - [swissprot-AllText:duchenne*] > omim
OMIM-Einträge zu denen man direkt von SwissProt-Einträgen mit Begriff "duchenne" gelangt
 - [swissprot-AllText:duchenne*] > pdb > omim
wie oben, doch SwissProt und OMIM müssen über PDB verlinkt sein
 - [swissprot-id:acha_human] > prosite > swissprot
Suche nach Eintrag "acha_human", Link nach Prosite (Protein Fam.) "neuronal acetylcholine receptors", Link zu Swissprot Einträgen => Sequenzen aus einer Proteinfamilie
 - [swissprot-id:gshr_caeel] > prodom > pdb
kein direkt Link von "gshr_caeel" zur PDB, aber über ProDom (Protein Domainen), Ergebnis homologe Proteine zu "gshr_caeel"

SRS Anfrage Sprache

- Einträge und Untereinträge
 - [swissprot-keywords:transmembrane] alle Einträge mit Keyword Transmembr.
 - [swissprot-ftkey:transmem] Menge von Untereinträgen mit Typ transmem
- Einträge und Untereinträge können über Links kombiniert werden
 - [swissprot-org:human] > [swissprot-ftkey:transmem]
Menge der Transmembransegmente in menschlichen Proteinen
 - [swissprot-org:human] < [swissprot-ftkey:transmem]
Menge der menschlichen Proteine mit Transmembransegmenten
 - [swissprot-ftkey:transmem] > parent
Konvertierung der Untereinträge zu den zugehörigen Einträgen
 - [swissprot-ftkey:transmem] > parent | [swissprot-key:transmembrane]
Einträge mit Transmembransegmenten oder mit dem Schlüsselwort "transmembrane"

SRS Anpassung

- Komplizierte Anfragen können vordefiniert werden
- Perl-ähnlich Sprache ICARUS
- Beispiel: Suche einen Swissprot-Eintrag nach Zugriffsnummer oder Beschreibung


```
$CannedQuery:[sampleQuery
prompt:Sample canned query
options:{
  $AppOpt:[ac prompt:'Access number' defStr:'Q1']
  $AppOpt:[des prompt:'Description' defStr:'cancer']
}]
queryStr:
@[swissprot-des:($des)*][swissprot-acc:($ac)*]"
```
- Das @ zeigt, dass die Anfrage zur Laufzeit bearbeitet wird.
- Datei mit Anfragedefinition wird ähnlich wie ein HTML Link in SRS-Seite eingebunden.

Aufruf von SRS aus HTML Seiten

- Wegtz kann als CGI Programm direkt als HTML Link aufgerufen werden
 - Einfaches Anzeigen von Einträgen
 - wget?-e+[embl-id:rnelas]
 - wget?-e+[{embl%20emblnew}-acc:X012345]
 - Die Option -e lässt wgetz volle Einträge anzeigen
 - Anzeigen von Mengen
 - wgetz?[embl-all:elastase]
 - wgetz?swissprot
 - wgetz?swissprot+-lv+30+-bv+31 (-lv Anzahl Einträge pro Seite, -bv erster Eintrag)
 - wgetz?swissprot+-lv+30+-bv+31+-view+SequenceSimple zeigt ZusatzInfos mit Hilfe eines vordef. Views
 - wgetz?swissprot+-lv+30+-bv+31+-view+SequenceSimple+-ascii+-rs+||+-cs+@@
- Wie oben aber als HTML Tabelle

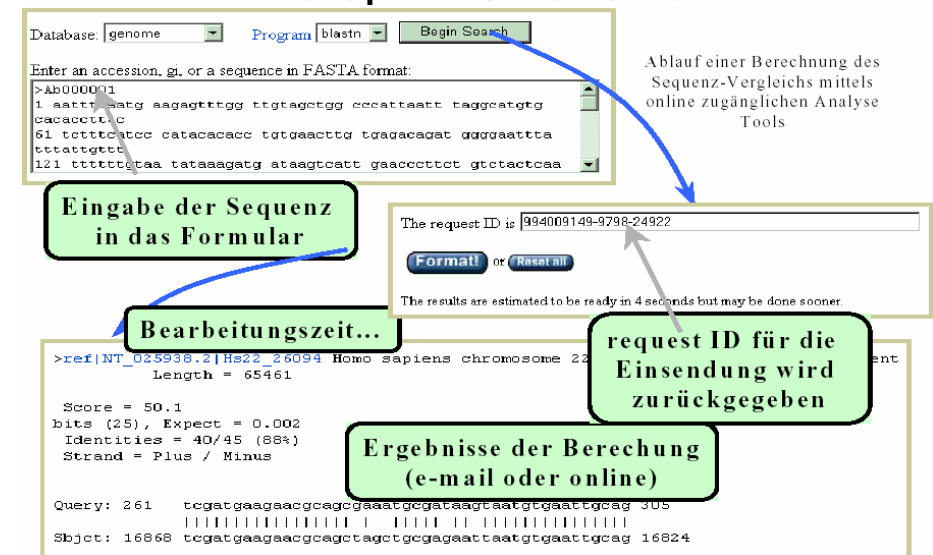
Schnittstellen für Analyse-Tools

- n Vielzahl an Analyseprogrammen für Verarbeitung von Bio-Daten, z.B.
 - CLUSTALX (Graphisches Tool für ClustalW multiple sequence alignment program)
 - FASTA
 - BLAST (Basic Local Alignment Search Tool)
 - Sacc3D (Structural Information for Yeast Proteins)
- n Installation auf lokalem Rechner vs. Remote-Zugang über Web-Interface
- n Installation auf lokalem Rechner: Effizient, aber Update-Problematik
- n Zugang über Web-Interface
 - Via Formulare und ftp/e-mail: Einfach zu realisieren, aber oft umständlich und wenig flexibel
 - Via API von Remote User Defined Functions (RUDFs): Flexibler, aber komplexer
 - Realisierung von RUDFs als Internet Functions

Eigenschaften von Internet Functions (IFs)

- n Kein Teil der Datenbank
- n Werden von einem externen System gestartet
- n Erreichbar über das Internet
- n Beinhalten Kommunikationsprotokoll zum Datenaustausch mit der Datenbank
- n Verschachtelungen mehrerer IFs beim Aufruf möglich

Beispiel-Szenario



Internet Function Definition Language (IFDL)

- n Erweiterung der SQL DDL
 - *Kein* Teil des SQL-Standards
 - Nicht verwechseln mit der *Independent Form Definition Language* (Abk. auch IFDL)
- n Definition für die Einbindung von IFs in SQL-Queries
- n IF-Definition: 5-Tupel
 - Funktionsname der aufzurufenden IF
 - URL der Funktion
 - Liste von Eingabeparametern mit Typen
 - Typ des Rückgabewertes
 - HTQL*-Wert

Hyper Text Query Language (HTQL)

- n Unterstützt Suche in Web-Dokumenten
- n Nutzung der Tags in HTML/XML-Dokumenten zur Navigation und Ausgabestrukturierung (ähnlich wie bei XPath, XQuery)
- n Dient im Zusammenhang mit IFs als Schnittstelle zw. DB und IFs
 - HTQL-Wert in IFDL Function Definition informiert über Position des Rückgabewertes im (generierten) Web-Rückgabedokument
 - IF extrahiert für weitere Verarbeitung Rückgabewert aus dem (generierten) Web-Dokument

Beispiel für IF-Ausführung (GenBank)

- n Vergleich von neuen, noch uncharakterisierten Gensequenzen (in Tabelle *local*) mit bekannten Sequenzen der Kenianischen Fruchtfliege
 - Nur solche Sequenzen von Interesse, die Ähnlichkeit von mind. 98 % zu geg. Sequenz aufweisen
 - Für Berechnung der Editierdistanz Verwendung der über Internet Schnittstelle zugänglichen Programms BLAST (<http://ncbi.nlm.nih.gov/blast/blast.cgi>)



Beispiel für IF-Ausführung (2)

- n Verwendung von zwei IFs: *blast*, *get_seq*
 - *blast* verwendet als Eingabe eine Sequenz der Tabelle *local* und liefert "requestID"-Wert (rein "technischer" Parameter für weitere Referenz auf Anfrage, vor allem wg. oft langer Laufzeit von *blast*)
 - Ergebnis von *blast* befindet sich nach Ausführung in von *Blast.cgi* erzeugter HTML-Datei im Eingabefeld des 1. form-Tags
 - Dieser Wert wird extrahiert und dient als Parameter für *get_seq*, dessen Ergebnis nach dem 2. pre-Tag der erzeugten HTML-Datei steht
 - Ergebnis von *get_seq* ist Tabelle, welche die ermittelten Ähnlichkeitsscores, Herkunft und Art enthält

```
define function blast
href "http://www.ncbi.nlm.nih.gov/blast/Blast.cgi"
parameters query varchar(10000)
results request_id varchar(40)
htql value: <form>.<input>;
```

```
define function get_seq
href "http://www.ncbi.nlm.nih.gov/blast/Blast.cgi"
parameters rid varchar(40)
results sequence varchar(10000)
htql value: <pre>.<pre>;
```

Beispiel für IF-Ausführung (3)

n SQL Query mit IF-Aufrufen

```
SELECT b.sequence
FROM (SELECT get_seq( blast( a.sequence ))
      FROM local as a) as b
WHERE b.organism = "Drosophila" AND b.source(country) = "Kenia"
      AND b.e-value >= 0.98
```

n Teile der Anfrage

- Führe für alle Sequenzen der DB-Tabelle *local* (hier *a*) die BLAST-Suche in GenBank durch
- Speichere die Ergebnisse in die Tabelle *b* ab und gib alle Sequenzen aus, die zur Kenianischen Fruchtfliege gehören und einen Ähnlichkeits-Score von mehr als 0.98 haben

Anwendung: LifeDB

n Biologisches Datenbanksystem (Prototyp der Mississippi State University)

n Unterstützt das Konzept der IFs

n Dreiteiliges Interface

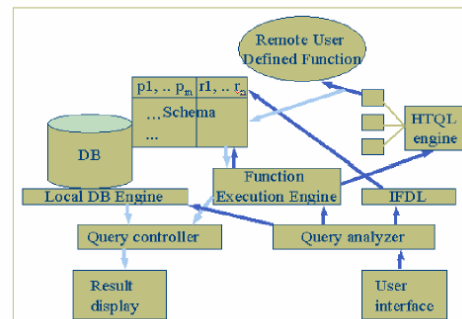
- Textfeld links oben für Eingabe der IFs und SQL-Anfragen
- Query-Ergebnisseite rechts
- Feld links unten zur Auflistung bereits getätigter Anfragen

The screenshot shows the LifeDB web interface. On the left, there is a text input field for queries. Below it, a list of queries is shown, including '345452 sts_code, GenbankID' and '1 queries refresh'. On the right, the 'Results' section displays the query '345452' and a table with columns 'sts_code' and 'GenbankID'. The table contains 5 records, with the first one being 'WT-5270 604845'. Below the table, there is a '5 records found refresh' link and a 'Return Home' link.

LifeDB Anfrageverarbeitung: Architektur

n Query Analyzer

- Parsen von Eingaben im Query-Eingabefeld (stand-alone IF-Aufrufe sowie SQL-Anfragen mit oder ohne eingebundene IF)
- Weiterleitung von SQL-Anfragen ohne eingebundene IF direkt an die lokale DB Engine
- Weiterleitung von IF-Aufrufen an IFDL Modul



n IFDL Modul

- Weiterleitung von eingebetteten IF an Function Execution Engine
- Verfügt über Metadaten-Tabelle in lokalen Datenbank mit Strukturinformationen über die definierten IF (diese wird von Function Execution Engine benötigt, um IF über HTQL-Engine einzubinden und auszuführen)

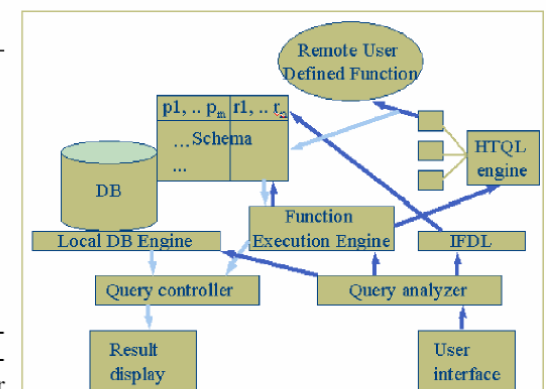
```
create table IFDLUDFS (
  function_name varchar(30),
  store_table_name varchar(30),
  address varchar(100),
  para_num number(2),
  para_name_list varchar(200),
  result_name_list varchar(200),
  result_html varchar(200))
```

IF-Metadaten-Tabelle

LifeDB Anfrageverarbeitung: Architektur (2)

n HTQL Engine

- Nutzt http zur Übermittlung der Eingabedaten an IF
- IF wird extern ausgeführt und legt Ergebnis in HTML-Seite ab
- HTQL Engine extrahiert Ergebnis anhand des HTQL-Wertes der IF-Definition
- Ergebnis wird in einer weiteren Tabelle in lokaler Datenbank abgelegt



n Query Controller führt SQL-Anfrage aus und verwendet dabei Ergebnistabelle aus der Ausführung der IF

n Schließlich wird Ergebnis dem Nutzer präsentiert

RUDF/IF: Zusammenfassung

- n API / CLI Ansatz zur Einbindung externer Funktionalität bei der Auswertung von Bio-Daten
- n Einbindung von RUDF / IFs innerhalb von SQL-Queries
- n Rückgabe der Ergebnisse in html-Format mit Extraktionsinformation (wo ist gewünschte Information im Rückgabe-Dokument?)
- n Vorteile: Umgehung lokaler Installation von externer Funktionalität
- n Nachteile: Proprietäre Formate und Ausführungsarchitektur
- n Weiter Informationen unter:
 - <http://www.cse.msstate.edu/~cly/EDI/index.html>
 - Chen, L.; Jamil, H. M.: Supporting Remote User Defined Functions in Heterogeneous Biological Databases. Proceedings IEEE International Conference on BIBE 2001: 144-152, 2001.
 - Chen, L.; Jamil, H. M.: On Using Remote User Defined Functions as Wrappers for Biological Database Interoperability; International Journal of Cooperative Information Systems (IJCIS), Special Issue on Data Management and Modeling Support in Bioinformatics, 12(2):161-195, 2003.

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 575

Deklarative Anfragesprachen in Bio-Datenbanken

- n Unterstützen (semi-)strukturierte Anfragen (i. allg. via Mengenoperationen)
- n Vertreter
 - SQL92 (relational), SQL:2003 (objekt-relational)
 - OQL (objektorientiert)
 - XPath / XQuery (XML-basiert)
- n Verwendbarkeit für Bio-Datenbanken?
 - Bio-Daten: Oft nur semi-strukturiert
 - Ggf. hoher Multimedia-Anteil: Text, Images (z.B. bei Genexpression), Verlaufskurven (z.B. bei medizinischen Daten)
- n Kaum Verwendung als "öffentliches" Bio-Interface, da zu komplex
- n Bio-Speziessprachen: BioSQL, PQL, ...

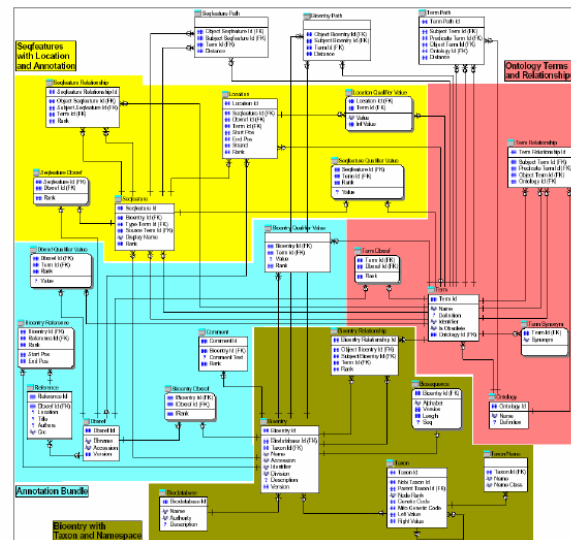
Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 576

Beispiel: BioSQL

- n Auf Basis von SQL92
- n Keine SQL-Spracherweiterung, sondern spezifische DDL-Ausprägung
- n OpenSource Projekt (<http://cvs.bioperl.org/>)
- n Relationale Schemata (Oracle, Postgres, mySQL)
- n EMBL & SwissProt
 - Ca. 20 SQL-Tabellen

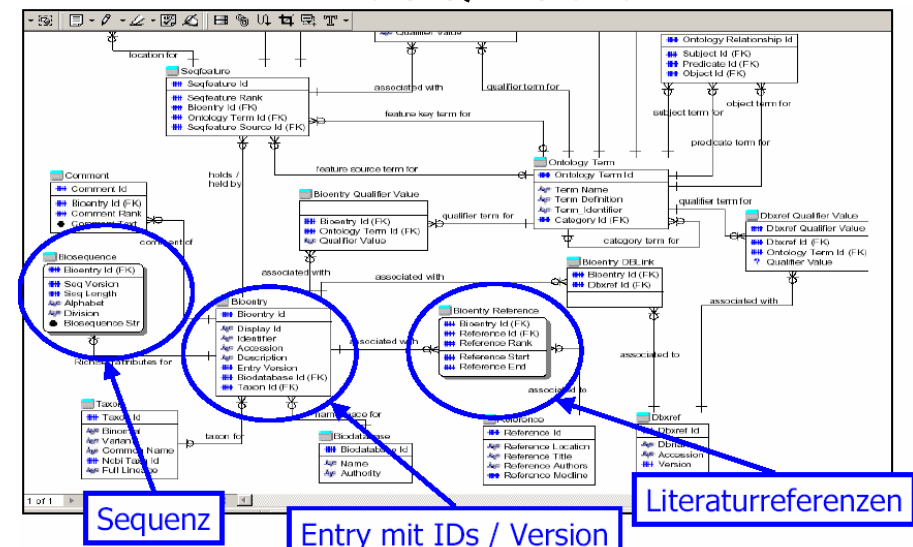


Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 577

BioSQL Core

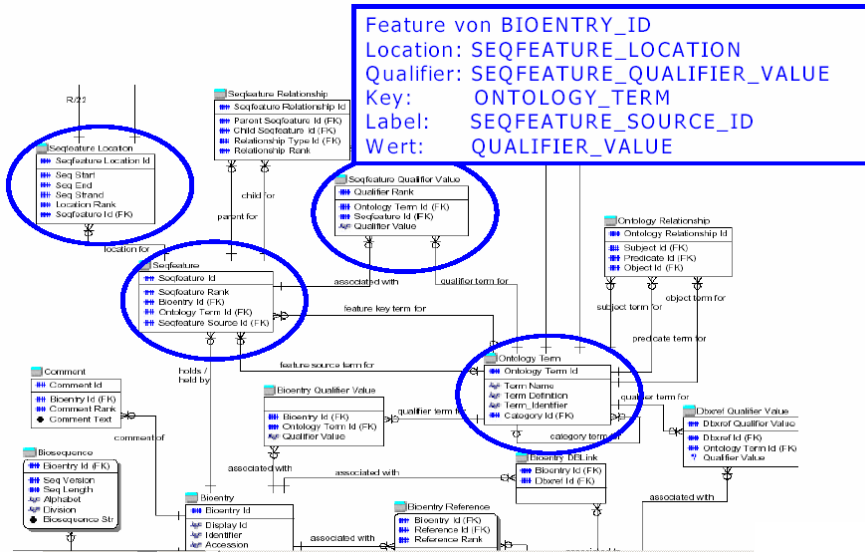


Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 578

BioSQL Feature



Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 579

BioSQL und BioPython

- BioSQL liefert API für mehrere Programmiersprachen
Python, Perl, Ruby, Java, ...
- Zugriff auf BioSQL DB aus Python

```
>>> from BioSQL import BioSeqDatabase
>>> server = BioSeqDatabase.open_database(driver = "MySQLdb", user = "chapmanb",
...                                     passwd = "biopython", host = "localhost", db = "bioseqdb")
>>> db = server.new_database("cold")

>>> from Bio import GenBank
>>> parser = GenBank.FeatureParser()
>>> iterator = GenBank.Iterator(open("cor6_6.gb"), parser)
>>> db.load(iterator)

6
>>> record = db.lookup(accession = "X62281")
>>> record.id
'X62281'
>>> record.name
'ATKIN2'
>>> record.description
'A.thaliana kin2 gene.'
```

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 580

DiscoveryLink

[L.M. Haas, P.M. Schwarz, P. Kodali, E. Kotlar, J. E. Rice und W. C. Swope, *DiscoveryLink: A system for integrated access to life sciences data sources*. IBM Systems Journal, 40(2):489-511, 2001.]

- kommerzielles System von IBM
- vollkommene Transparenz für Anwender
- Anwender denkt: Ich arbeite mit einer Datenquelle!
- Vorgänger war GARLIC-Projekt:
Integration von heterogenen Quellen mit
Spezifikation eines integrierten Schemas und unter
Nutzung von Wrappern
- Ankopplung von Analysewerkzeugen ist primäres Ziel
von Discovery Link
- FDBMS-Ansatz

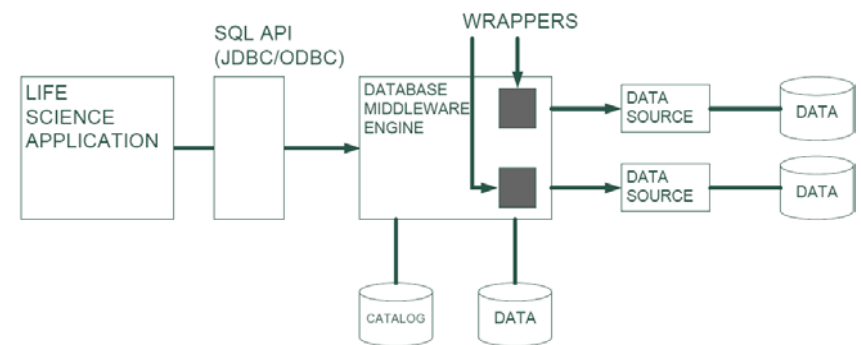
Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 581

DiscoveryLink

Architektur:



Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 582

DiscoveryLink

- Wrapper sind in C++ implementiert und werden dynamisch geladen

```
CREATE WRAPPER ChemWrapper LIBRARY
'libchemdb.a'
```

- Definition eines Servers auf den mittels Wrapper zugegriffen wird

```
CREATE SERVER Chem-HTS WRAPPER ChemWrapper
OPTIONS(NODE 'hts1.bigpharma.com',
PORT '2003', VERSION '3.2b')
```

- Schemadefinition eines Wrappers

Protein_Sequence Wrapper Schema	Assay Wrapper Schema	Molecule Wrapper Schema
<pre>CREATE NICKNAME PROTEINS { PROTEIN_ID VARCHAR(30) NOT NULL, NAME VARCHAR(60), FAMILY VARCHAR(256), DISEASES VARCHAR(256) } SERVER PROTEINDB</pre>	<pre>CREATE NICKNAME ASSAYS { COMPOUND_ID VARCHAR(10) NOT NULL, SCREEN_NAME VARCHAR(30) NOT NULL, IC50 DECIMAL(12,10) } SERVER ORACLE12</pre>	<pre>CREATE NICKNAME COMPOUNDS { COMPOUND_ID VARCHAR(10) NOT NULL, STRUCTURE LONG VARCHAR, MOL_WT DECIMAL(10,6), LOGP DECIMAL(10,2) } SERVER CHEM-HTS CREATE FUNCTION MAPPING FOR SIMILARITY(LONG VARCHAR, LONG VARCHAR) RETURNS FLOAT SERVER CHEM-HTS</pre>

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 583

DiscoveryLink

- Anfragen werden mittels SQL formuliert

```
SELECT a.compound_id, a.IC50, p.name,
       c.structure
FROM Assays a, Proteins p, Compounds c
WHERE a.screen_name = p.protein_id
      AND a.compound_id = c.compound_id
      AND p.family LIKE '%serotonin%'
      AND a.IC50 < 1E-8
```

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 584

DiscoveryLink

- Anfragen werden optimiert
- Erste Phase: Zugriff auf einzelne Tabellen optimieren
- Predikate an Datenquellen delegieren

RemoteQuery	RemoteQuery	RemoteQuery
Tables: Assays	Tables: Compounds	Tables: Proteins
Columns: compound_id IC50 screen_name	Columns: compound_id structure	Columns: protein_id name
Predicates: IC50 < 1E-8	Predicates: (None)	Predicates: family LIKE '%serotonin%'

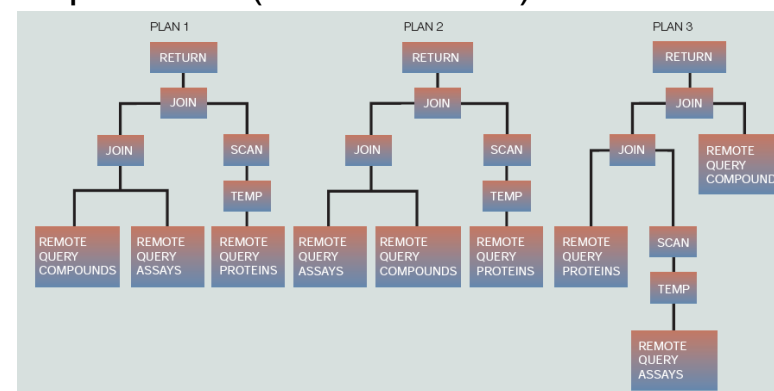
Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 585

DiscoveryLink

- Zweite Phase: Joins zwischen Tabellen optimieren (links -> rechts)



Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 586

DiscoveryLink

• Komplexere Anfragen

```
SELECT a.compound_id., a.IC50, p.name,  
       c.mol_wt, c.logP, c.structure,  
       similarity(c.structure,  
                 :KETANSERIN_MOL) AS rank  
FROM Assays a, Proteins p, Compounds c  
WHERE a.screen_name = p.protein_id  
      AND a.compound_id = c.compound_id  
      AND p.family LIKE '%serotonin%'  
      AND a.IC50 < 1E-8  
      AND c.mol_wt BETWEEN 375 AND 425  
      AND c.logP BETWEEN 4 AND 5  
ORDER BY rank
```

DiscoveryLink

- FDBMS -> integriertes Schema
- Zugriff auf Originaldaten durch Views
- kommerzielles Produkt
Kostet sehr viel Geld!!! ☹
- Plattformunabhängigkeit
- Internetfähigkeit
- lesendes SQL
- JDBC-, ODBC-Treiber
- verschiedenste Ausgabeformate
- keine Einbindung eigener Datenquellen, immer Wrapper-Implementation notwendig
- Informationsfusion wird unterstützt

SQL92: Bewertung für Bio-Datenbanken

n Vorteile

- Breite Verwendung in kommerziellen Datenbank-Systemen
- Gute Einbindung in Programmierspachen
- Mächtig, effizient, stabil

n Nachteile

- Für Anfrage muss zugrundeliegendes Datenschema im Detail bekannt sein
- Eingeschränkte Unterstützung von semi-strukturierten Daten und Multimedia-Objekten

n Ausweg: SQL:2003 ??

Typsystem von SQL:2003

n Erweiterbares Typkonzept

- vordefinierte Datentypen
- konstruierte Typen (Konstruktoren): REF, Tupel-Typen (ROW-Typ), Kollektionstyp ARRAY
- benutzerdefinierte Datentypen (User-Defined Types, UDT): Distinct Types und Structured Types

n Definition von UDTs unter Verwendung von vordefinierten Typen, konstruierten Typen und vorher definierten UDTs

n UDTs unterstützen u.a.

- Kapselung
- Vererbung (Subtypen)

n alle Daten werden weiterhin innerhalb von Tabellen gehalten

- Definition von Tabellen auf Basis von strukturierten UDTs möglich
- Bildung von Subtabellen (analog zu UDT-Subtypen)

n neue vordefinierte Datentypen

- Boolean (Werte: TRUE, FALSE, UNKNOWN)
- Large Objects (Binärdarstellung bzw. Texte), Multimedia-Objekte

Large Objects

- n 3 neue Datentypen:
 - **BLOB** (Binary Large Object)
 - **CLOB** (Character Large Object): Texte mit 1-Byte Character-Daten
 - **NCLOB** (National Character Large Objects): ermöglicht 2-Byte Character-Daten für nationale Sonderzeichen (z.B. Unicode)
- n Verwaltung großer Objekte im DBS (nicht in separaten Dateien)
 - umgeht große Datentransfers und Pufferung durch Anwendung; Zugriff auf Teilbereiche
- n Beispiel Genexpression (stark vereinfacht):

```
CREATE TABLE GenExpression-Data
  (Exp-ID      INTEGER,
   Lab         VARCHAR (40),
   Annotation   CLOB (75K),
   Exp-Image    BLOB (12M) )
```

- n indirekte Verarbeitung großer Objekte über Locator-Konzept (ohne Datentransfer zur Anwendung) (SQL/MED; IBM DB2 DataLink)

Large Objects (2)

- n unterstützte Operationen
 - Suchen und Ersetzen von Werten (bzw. partiellen Werten)
 - LIKE-Prädikate, **CONTAINS**, **POSITION**, **SIMILAR**
 - Konkatenation ||, SUBSTRING, LENGTH, IS [NOT] NULL ...
- n einige Operationen sind auf LOBs nicht möglich
 - Schlüsselbedingung
 - Kleiner/Größer-Vergleiche
 - Sortierung (ORDER BY, GROUP BY)
- n Beispielanfrage Genexpression

CREATE TABLE GenExpression-Data (Exp-ID INTEGER, Lab VARCHAR (40), Annotation CLOB (75K), Exp-Image BLOB (12M))	SELECT Exp-Id, Exp-Image FROM GenExpression-Data WHERE Annotation.Contains("E. Coli")
---	--

SQL/MM (Multimedia and Applications Packages)*

- n Multimedia/Anwendungs-Erweiterungen auf Basis von SQL:2003
 - Full Text (Datentypen und Funktionen zur Volltextsuche), z.B. für Bio-Annotationen
 - Spatial (räumliche Daten), z.B. für Tertiär- und Quartärstruktur von Proteinen
 - Still Image (unbewegte Bilder), z.B. für Genexpression-Images
 - Data Mining
- n SQL/MM Full Text
 - Deutlich leistungsfähiger als CLOBs
 - Neue Datentypen FullText und FT_Pattern (Suchmuster)
 - Suchausdruck: Contains
 - Ähnlichkeitssuche: Rank (Grad der Übereinstimmung)
 - Kontextsuche: in same sentence as, in same paragraph as
 - linguistische Suche: stemmed form of

SQL/MM (2)

- n SQL/MM Still Image
 - Datentyp SI_StillImage für digitale Rasterbilder (interne Speicherung als BLOB)
 - Methoden für Formatkonversionen, Skalierung, Rotation, Ausschnittbildung, Farbbestimmung ...
- n Unterstützung der Suche durch Feature-Datentypen:
 - SI_AverageColor (durchschnittlicher Farbwert): Aufsummierung der einzelnen Farbwerte jedes Pixels (z.B. rot, grün, blau); Division durch Anzahl der Pixel
 - SI_ColorHistogram (Farbverteilung),
 - SI_PositionalColor (Farbverteilung bezüglich Bildlage): Bild wird in n mal m Rechtecke eingeteilt; Berechnung des Farbdurchschnittswertes eines Bereich analog zu SI_AverageColor
 - SI_Texture (Angaben zu Auflösung, Kontrast etc.)
 - Suchmethode SI_Score (liefert Maß der Übereinstimmung)
- n SQL/MM Spatial
 - zunächst nur 0-, 1- und 2-dimensionale Objekte (Punkte, Linien, Flächenformen)
 - Bereitstellung standardisierter Typhierarchien wie ST_Geometry (Subtypen ST_Point, ST_Curve, ST_MultiPolygon) und ST_SpatialRefSys (räumliche Referenzsysteme)
 - Methoden zur Erzeugung von Geo-Objekten und Berechnungen (intersect, overlap, area ...)

SQL/MM (3)

- n Beispiel Genexpression (stark vereinfacht)

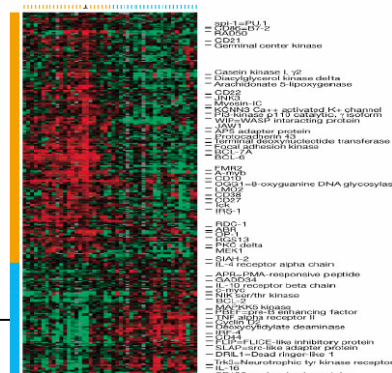
```
CREATE TABLE GenExpression-Data
(Exp-ID      INTEGER,
Lab          VARCHAR (40),
Annotation   FULLTEXT,
Exp-Image    SL_STILLIMAGE)
```

```
SELECT Exp-Id, Exp-Image
FROM   GenExpression-Data
WHERE  Annotation.Contains("T-Lymphocyte" in same paragraph as "Liver" )
      AND P.Annotation.Rank ("Hon-Hodgkin-Lymphoma") > 0.25
      AND WHERE 1.2 < SI_InitFeatureList
      (SI_findTexture(Sample-Exp-Image), 0.8)
      .SI_Append
      (SI_findPositionalColor(Sample-Exp-Image), 0.2)
      .SI_Score(Exp-Image)
```

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 595



SQL:2003: Bewertung

- n SQL:2003-Standardisierung
 - Kompatibilität mit existierenden SQL-Systemen
 - Objektorientierung
 - Unterstützung von semi-strukturierten Daten und Multimedia
 - SQL/MED: DBS-Einbindung von externen Daten / Dateien
- n Daher: Als Anfragesprache für Biodaten prinzipiell geeignet
- n Aber: Bzgl. Datenschema muss a priori immer noch bekannt sein, welche Teile wie strukturiert und welche unstrukturiert sind (im Ggs. zu XML)

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 596

XML-basierte Anfragesprachen

- n XQuery: W3C-Standardisierungsprozeß für einheitliche XML-Anfragesprache
- n abgeleitet von vorangegangenen proprietären XML-Anfragesprachen (Quilt, XPath, XQL, XML-QL, ...) sowie SQL und OQL
- n funktionale Anfragesprache
- n komplexe Pfadausdrücke (basierend auf XPath 2.0), Funktionen, konditionale und quantifizierte Ausdrücke, Ausdrücke zum Testen/Modifizieren von Datentypen, Elementkonstruktoren
- n FLWOR-Syntax (For ... Let ... Where ... Order By ... Return)

XQuery: FLWOR

- n Generelle Vorgehensweise
 - Dokumentzugriff mit doc(url)
 - Knotennavigation mit XPath 2.0 (z.B. //book oder /book/author[Name = "Rahm"])
 - Variablenbindung im Kontext existierender Bindungen
 - Operationen auf den Knotenmengen
 - Erzeugung neuer Knoten

- n FLWOR-Ausdrücke

```
FLWOR-expr ::= (FOR-expr | LET-expr)+
              WHERE-expr? ORDERBY-expr? RETURN-expr
FOR-expr   ::= for $var in expr (, $var in expr)*
LET-expr   ::= let $var := expr (, $var := expr)*
WHERE-expr ::= where expr
ORDERBY-expr ::= (order by | stable order by) OrderSpecList
OrderSpecList ::= OrderSpec (, OrderSpec)*
OrderSpec   ::= expr OrderModifier
OrderModifier ::= (ascending | descending)? (empty greatest | empty least)? (collation StringLiteral)?
RETURN-expr ::= return expr
```

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 597

Sommersemester 2005/06

Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Seite 598

XQuery: FLWOR (2)

n LET-Ausdruck

- bindet Menge von Werten an Variable
- Menge wird geschlossen an Variable gebunden

```
let $n := //Book/Author/Name
return $n
```

n FOR-Ausdruck

- für jedes Element der Ergebnismenge erfolgt Bindung an Variable (Iteration über die Menge)
- Beispiel liefert gleiches Ergebnis wie bei LET, jedoch wird es anders abgearbeitet:
 - \$n wird jeweils an Elemente der Sequenz von Autorennamen gebunden (für jeden Namen genau einmal)
 - RETURN wird für jede Bindung einmal ausgeführt
 - Zwischenergebnisse werden zu Gesamtergebnis zusammengefaßt

```
for $n in //Book/Author/Name
return $n
```

XQuery: Bio-Beispiel

n Ausgangsschema

```
<!ELEMENT seq (dbref, family?, full_seq, annotation*) >
<!--ATTLIST seq id ID #REQUIRED
name CDATA #IMPLIED
length CDATA #IMPLIED -->

<!ELEMENT dbref (database, unique_id, version?)>
<!--ELEMENT database (#PCDATA) >
<!--ELEMENT unique_id (#PCDATA) >
<!--ELEMENT version (#PCDATA) >
<!--ELEMENT family (#PCDATA) >
<!--ELEMENT full_seq (#PCDATA) >
<!--ATTLIST full_seq type (dna | rna | aa) #REQUIRED-->
<!--ELEMENT annotation (#PCDATA) >
```

DTD

Daten-satz

```
<seq id="my_seq" name="NUCLEAR RIBONUCLEOPROTEIN">
  <dbref>
    <database>SWISS-PROT</database>
    <unique_id>P09651</unique_id>
  </dbref>
  <family>RIBONUCLEASEN TYPE 15</family>
  <full_seq type="aa">
    SKSESPKEPEQLRKLFIGGLSFETTTDESLSRSHFEQWGLTDCVVMRDPTKRSRGFGVYATVEEVDAMNARPH
    KVDGRVVEPKRAVSRSDSQRGAHLTVKKI FVGGIKEDTEHHLRDYFEQYKIEVIEIMTDRSGGKRGFAFVTF
    DDHDSVDKIVIQYHTVNGHNCVEVKALSKQEMASASSSQRRSGSGNFGGGRGGGFGGNDNFGRGNFSGRGGFG
    GSRGGGGYGGSGDGYNGFGNDGGYGGGGPGYSGGSRGYSGGGQGYGNQGGGGYGGSGSYDSYNNGGGRGFSGGGGSN
    FGGGGSYNDFGYNNQSSNFGPMKGGNFGGRSSGPGYGGGGQYFAKPRNQGGYGGSSSSSYSGSGRRF
  </full_seq>
  <annotation> This protein binds to RNA and ... </annotation>
</seq>
```

XQuery: Bio-Beispiel (2)

```
<!--ELEMENT seq (dbref, family?, full_seq, annotation*) >
<!--ATTLIST seq id ID #REQUIRED
name CDATA #IMPLIED
length CDATA #IMPLIED -->

<!--ELEMENT dbref (database, unique_id, version?)>
<!--ELEMENT database (#PCDATA) >
<!--ELEMENT unique_id (#PCDATA) >
<!--ELEMENT version (#PCDATA) >
<!--ELEMENT family (#PCDATA) >
<!--ELEMENT full_seq (#PCDATA) >
<!--ATTLIST full_seq type (dna | rna | aa) #REQUIRED-->
<!--ELEMENT annotation (#PCDATA) >
```

- Liste alle Elemente, die etwas zu "RIBONUCLE..." enthalten

```
doc("http://data.bio-server.bio-university.edu/biodata.xml")//*[contains(text(), 'RIBONUCLE')]
```

- Liste alle Aminosäuresequenzen mit Namen auf, welche die Teilsequenz "SFETTTDESLRSH" enthalten, in alphabetischer Reihenfolge

```
for $s in doc("http://.../biodata.xml")//seq
where $s/full_seq/@type = aa and
      $s/full_seq[contains(text(), 'SFETTTDESLRSH')]
order by $s/@name
return $s/@name
```

XQuery: Bio-Beispiel (3)

```
<!--ELEMENT seq (dbref, family?, full_seq, annotation*) >
<!--ATTLIST seq id ID #REQUIRED
name CDATA #IMPLIED
length CDATA #IMPLIED -->

<!--ELEMENT dbref (database, unique_id, version?)>
<!--ELEMENT database (#PCDATA) >
<!--ELEMENT unique_id (#PCDATA) >
<!--ELEMENT version (#PCDATA) >
<!--ELEMENT family (#PCDATA) >
<!--ELEMENT full_seq (#PCDATA) >
<!--ATTLIST full_seq type (dna | rna | aa) #REQUIRED-->
<!--ELEMENT annotation (#PCDATA) >
```

- Geben Sie zu jeder Proteinfamilie die durchschnittliche Länge der Proteinsequenzen aus

```
for $f in distinct-values(doc("http://.../biodata.xml")//family)
where $f/full_seq/@type = aa
let $a := avg(doc("http://.../biodata.xml")//seq[family = $f/full_seq/string-length( text() )])
return
  <family>
    <name> { $f/text() } </name>
    <avglength> { $a } </avglength>
  </family>
```

XML/XQuery: Bewertung

- n Standardisiert, objektorientierte Mächtigkeit
- n Als Anfragesprache für Bio-Daten prinzipiell geeignet
 - Gute Unterstützung von semi-strukturierten Daten
 - Anfragen auch mit sehr wenig Metawissen über Struktur der Bio-Daten möglich
- n Nachteile
 - Noch unklare Situation bzgl. Produkten (Native XML-DBMS vs OR-DBMS mit XML-Modulen)
 - Keine Unterstützung von Multimedia-Objekten