Bioinformatics Resources - Structural Resources / SQL -

Lecture & Exercises
Prof. B. Rost, Dr. L. Richter, J. Reeb
Institut für Informatik I12
## Preliminary Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apr. 28th</td>
<td>Intro, General Overview (1. sh.)</td>
</tr>
<tr>
<td>May 5th</td>
<td>Sequence Databases (2. sh.)</td>
</tr>
<tr>
<td>May 12th</td>
<td>Sequence Databases (3. sh.)</td>
</tr>
<tr>
<td>May 19th</td>
<td>Structure Databases (4. sh.)</td>
</tr>
<tr>
<td>May 26th</td>
<td>No Lecture</td>
</tr>
<tr>
<td>Jun 2nd</td>
<td>SQL (5. sh.)</td>
</tr>
<tr>
<td>Jun 9th</td>
<td>SQL, NoSql (6. sh)</td>
</tr>
<tr>
<td>Jun 28th</td>
<td>No Lecture</td>
</tr>
<tr>
<td>Jun 16th</td>
<td>No Lecture</td>
</tr>
<tr>
<td>Jun 23rd</td>
<td>NoSql 2 (7.sh.)</td>
</tr>
<tr>
<td>Jun 30th</td>
<td>MongoDB, JavaScript (8.sh.)</td>
</tr>
<tr>
<td>Jul 7th</td>
<td>Node.js Applications (9.sh.)</td>
</tr>
<tr>
<td>Jul 14th</td>
<td>PredictProtein</td>
</tr>
<tr>
<td>Jul 21st</td>
<td>Wrap Up, Q&amp;A</td>
</tr>
<tr>
<td>Jul 28th</td>
<td>Exam</td>
</tr>
</tbody>
</table>

* These exercises can earn you a bonus
Orga - Exam Date

- Exam scheduled for Friday, Jul 28th
- Time: 16:30-18:00
- Room: MW 0350 Egbert-von-Hoyer Lecture Hall (Mechanical Engineering Building)
- Registration is MANDATORY
- so far 6 students registered
Secondary Databases

- Databases which digest and structure data from primary databases
- Not always “true” database systems
- SCOP/CATH
- PFAM
- PROSITE
Classification of Structures: CATH-Gene3D / SCOP

- came up in the middle of the 1990s
- both are quite similar
- aim: organize the protein structures available in PDB, based on single domains
- hierarchical system (roughly):
  - secondary structure content
  - fold
  - super families
  - families

BioinfRes SoSe 17
SCOP: a Structural Classification of Proteins

- fully manually curated, driven by expert analysis
- associated with the ASTRAL compendium
- latest news: SCOPe (UC Berkeley), SCOP2 (MRC Lab Mol Biol, Cambridge, UK)
SCOP: a Structural Classification of Proteins

Hierarchical Level

1. Classes: Consider secondary structure composition (all α, all β, α/β, α+β, multi-domain, membrane/cell surface/peptides,...)

2. Fold: Shape of a domain. Proteins of the same fold have the same major secondary structure elements in the same arrangement with the same topological features

3. Superfamily: Groups of domain which have at least a distant common ancestor
Hierarchical Level

5. Family: Groups within superfamilies with a more recent common ancestor (>30% sequences identity or >15% seq.id. plus same function

6. Protein domain: Groups within families, essentially the same protein (isoform, the same protein but from different species)

7. Species: Protein domains according to species

8. Domain: the single domain
Development starting from year 2000

Timeline of SCOP(e) releases

taken from http://scop.berkeley.edu/help/ver=2.06#scopchanges
taken from http://scop.berkeley.edu/help/ver=2.06#scopchanges
CATH - Faces

taken from http://www.tgac.ac.uk/scientific-advisory-board/

taken from http://www.ebi.ac.uk/about/people/janet-thornton
Publications


CATH

- semi-automatic procedure for deriving a novel hierarchical classification of protein domain structures

- four main levels:
  - C: protein class, mainly secondary structure composition of each domain
  - A: architecture, summarizes shapes based on orientation of secondary structure elements
  - T: topology, sequential connectivity is considered
  - H: homologous superfamily, high similarity with similar functions, evolutionary relationship assumed
from Structure 15, August 1997, 5:1093–1108
http://biomednet.com/elecref/0969212600501093

Annual increase in the numbers of protein domain structures in the PDB (top plot, [11,12]). The lower lines show the numbers of identical families (H-level, 100% sequence identity between structures within the family and 100% overlap), non-identical families (N-level, > 95% sequence identity, 85% overlap), sequence families (S-level, > 35% sequence identity, 60% overlap), homologous superfamilies (H-level, > 25% sequence identity, SSAP >80 and 60% overlap), and topological or fold families (T-level, SSAP >70), where SSAP is a structural comparison score.
Current Release

- CATHDB version: 4.0
- 235,000 domain
- 25 mio protein predictions
- new:
  - improved prediction of functional families
  - current putative domain assignments (CATH-B)
  - CATH-40: a non-redundant set of CATH domains for homolgy benchmarking experiments (<40% seq. id with 60% overlap)
- http://www.cathdb.info/wiki/doku/?id=release_notes#cath_release_notes
Numbering Scheme

- C: 1,2,3,4 (alpha, beta, alpha/beta, none) (4)
- A: same architecture, different topology (40)
- T: Topology (connection of secondary structure elements) (1373)
- H: Homology (families) (2737)
The numbers of fold families (T-level), homologous superfamilies (H-level) and domain structures in different architectures are shown for the mainly α, mainly β and α–β classes.

<table>
<thead>
<tr>
<th>Class</th>
<th>Architecture</th>
<th>Number of T-levels</th>
<th>Percentage of all T-levels*</th>
<th>Number of H-levels</th>
<th>Percentage of all H-levels*</th>
<th>Number of domains</th>
<th>Percentage of all domains*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mainly α</td>
<td>Non-bundle</td>
<td>86</td>
<td>17.03</td>
<td>93</td>
<td>14.42</td>
<td>1455</td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>Bundle</td>
<td>34</td>
<td>6.73</td>
<td>39</td>
<td>6.05</td>
<td>226</td>
<td>2.80</td>
</tr>
<tr>
<td></td>
<td>Few SS</td>
<td>25</td>
<td>4.95</td>
<td>25</td>
<td>3.88</td>
<td>112</td>
<td>1.39</td>
</tr>
<tr>
<td>Mainly β</td>
<td>Ribbon</td>
<td>17</td>
<td>3.37</td>
<td>17</td>
<td>2.64</td>
<td>114</td>
<td>1.41</td>
</tr>
<tr>
<td></td>
<td>Single sheet</td>
<td>5</td>
<td>0.99</td>
<td>6</td>
<td>0.93</td>
<td>56</td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td>Roll</td>
<td>6</td>
<td>1.19</td>
<td>6</td>
<td>0.93</td>
<td>55</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td>Barrel</td>
<td>22</td>
<td>4.36</td>
<td>29</td>
<td>4.50</td>
<td>861</td>
<td>10.66</td>
</tr>
<tr>
<td></td>
<td>Clam</td>
<td>1</td>
<td>0.20</td>
<td>1</td>
<td>0.16</td>
<td>1</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>Sandwich</td>
<td>21</td>
<td>4.16</td>
<td>43</td>
<td>6.67</td>
<td>1236</td>
<td>15.30</td>
</tr>
<tr>
<td></td>
<td>Distorted sandwich</td>
<td>14</td>
<td>2.77</td>
<td>14</td>
<td>2.17</td>
<td>83</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td>Trefoil</td>
<td>1</td>
<td>0.20</td>
<td>4</td>
<td>0.62</td>
<td>49</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>Orthogonal prism</td>
<td>1</td>
<td>0.20</td>
<td>1</td>
<td>0.16</td>
<td>4</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>Aligned prism</td>
<td>1</td>
<td>0.20</td>
<td>2</td>
<td>0.31</td>
<td>3</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>Four-propeller</td>
<td>1</td>
<td>0.20</td>
<td>1</td>
<td>0.16</td>
<td>3</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>Six-propeller</td>
<td>1</td>
<td>0.20</td>
<td>1</td>
<td>0.16</td>
<td>37</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td>Seven-propeller</td>
<td>2</td>
<td>0.40</td>
<td>2</td>
<td>0.31</td>
<td>11</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>Eight-propeller</td>
<td>1</td>
<td>0.20</td>
<td>1</td>
<td>0.16</td>
<td>2</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>Two-solenoid</td>
<td>2</td>
<td>0.40</td>
<td>3</td>
<td>0.47</td>
<td>5</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>Three-solenoid</td>
<td>1</td>
<td>0.20</td>
<td>1</td>
<td>0.16</td>
<td>1</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>5</td>
<td>0.99</td>
<td>5</td>
<td>0.78</td>
<td>104</td>
<td>1.29</td>
</tr>
<tr>
<td>α–β</td>
<td>Roll</td>
<td>24</td>
<td>4.75</td>
<td>33</td>
<td>5.12</td>
<td>469</td>
<td>5.81</td>
</tr>
<tr>
<td></td>
<td>Barrel</td>
<td>8</td>
<td>1.58</td>
<td>20</td>
<td>3.10</td>
<td>365</td>
<td>4.52</td>
</tr>
<tr>
<td></td>
<td>Two-layer sandwich</td>
<td>77</td>
<td>15.25</td>
<td>112</td>
<td>17.36</td>
<td>957</td>
<td>11.85</td>
</tr>
<tr>
<td></td>
<td>Three-layer (αβα) sandwich</td>
<td>78</td>
<td>15.45</td>
<td>115</td>
<td>17.83</td>
<td>1396</td>
<td>17.28</td>
</tr>
<tr>
<td></td>
<td>Three-layer (ββα) sandwich</td>
<td>3</td>
<td>0.59</td>
<td>3</td>
<td>0.47</td>
<td>11</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>Four-layer sandwich</td>
<td>4</td>
<td>0.79</td>
<td>4</td>
<td>0.62</td>
<td>12</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>Box</td>
<td>1</td>
<td>0.20</td>
<td>1</td>
<td>0.16</td>
<td>2</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>Horseshoe</td>
<td>1</td>
<td>0.20</td>
<td>1</td>
<td>0.16</td>
<td>1</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>34</td>
<td>6.73</td>
<td>34</td>
<td>5.27</td>
<td>253</td>
<td>3.13</td>
</tr>
<tr>
<td></td>
<td>Few SS</td>
<td>14</td>
<td>2.77</td>
<td>14</td>
<td>2.17</td>
<td>96</td>
<td>1.19</td>
</tr>
<tr>
<td>Few SS</td>
<td>Irregular</td>
<td>14</td>
<td>2.77</td>
<td>14</td>
<td>2.17</td>
<td>98</td>
<td>1.21</td>
</tr>
</tbody>
</table>
Pfam

- current version is 31.0, March 2017, 16712 families in 604 clans
- hosted by the EBI
Pfam

- Pfam-A: curated seed alignment derived from Pfamseq (UniProtKB based), profile HMMs for the seed alignment, full alignment with all HMM detected sequences
- Pfam-B: un-annotated, automatically generated from non-redundant cluster from ADDA
- focuses on single domains
Terms

- **Family**: collection of related protein regions
- **Domain**: structural unit
- **Repeat**: shot unit which is unstable in isolation but forms a stable structure when found in multiple copies
- **Motif**: short unit found outside globular domains
- **Clans**: related group of Pfam entries based on similarity in sequence, structure of profile-HMM
Site organisation

Clan

Proteome

Structure

Sequence

Family
Pfam Numbers (rel. 31)

- 16712 Pfam-A families
- 36% of the families are classified into 604 clans
- the Pfam-A release matches 73% of the 26.7 Mio sequences in the corresponding UniProt reference proteome database
- coverage of 90.5% of SwissProt human
- use of jackhmmer (from HMMER3 package)
- consider CATH and PDB
### Table 1. The reduction in size of RP versus full alignments

<table>
<thead>
<tr>
<th>Family identifier (accession)</th>
<th>Seed</th>
<th>Full</th>
<th>RP75</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC_tran (PF00005)</td>
<td>55</td>
<td>363 409</td>
<td>26% (93 265)</td>
</tr>
<tr>
<td>COX1 (PF00115)</td>
<td>94</td>
<td>254 351</td>
<td>1% (2006)</td>
</tr>
<tr>
<td>zf-H2C2_2 (PF13465)</td>
<td>163</td>
<td>227 898</td>
<td>61% (138 033)</td>
</tr>
<tr>
<td>WD40 (PF00400)</td>
<td>1804</td>
<td>193 252</td>
<td>65% (125 805)</td>
</tr>
<tr>
<td>MFS_1 (PF07690)</td>
<td>195</td>
<td>181 668</td>
<td>30% (55 719)</td>
</tr>
<tr>
<td>RVT_1 (PF00078)</td>
<td>152</td>
<td>172 360</td>
<td>5% (8257)</td>
</tr>
<tr>
<td>BPD_transp_1 (PF00528)</td>
<td>81</td>
<td>156 339</td>
<td>23% (36 523)</td>
</tr>
<tr>
<td>Response_reg (PF00072)</td>
<td>57</td>
<td>151 337</td>
<td>29% (44 329)</td>
</tr>
<tr>
<td>GP120 (PF00516)</td>
<td>24</td>
<td>146 453</td>
<td>N/A</td>
</tr>
<tr>
<td>HATPase_c (PF02518)</td>
<td>659</td>
<td>129 386</td>
<td>28% (36 085)</td>
</tr>
</tbody>
</table>

From release 27

BioinfRes SoSe 17
String

- **Protein Interaction Networks:** "STRING is a database of known and predicted protein-protein interactions. The interactions include direct (physical) and indirect (functional) associations; they stem from computational prediction, from knowledge transfer between organisms, and from interactions aggregated from other (primary) databases."

- 2031 organisms
- 9.6 mio proteins
- 1,380 mio interactions
String

- Protein Interaction Networks
- 2031 organisms
- 9.6 mio proteins
- 1,380 mio interactions
### Number of organisms

<table>
<thead>
<tr>
<th>Organism</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bacteria</td>
<td>1'678</td>
</tr>
<tr>
<td>Eukaryotes</td>
<td>238</td>
</tr>
<tr>
<td>Archaea</td>
<td>115</td>
</tr>
</tbody>
</table>

**Total organisms:** 2'031

### Number of proteins

9'643'763 proteins

### Number of interactions (by confidence level)

- 25'914'693 interactions at highest confidence (score >= 0.900)
- 71'673'028 interactions at high confidence or better (score >= 0.700)
- 320'182'220 interactions at medium confidence or better (score >= 0.400)
- 1'380'838'440 interactions at low confidence or better (score >= 0.150)

**Total interactions:** 1'380'838'440

### Top organisms (by query volume)

1. Homo sapiens
2. Mus musculus
3. Escherichia coli
4. Arabidopsis thaliana
5. Caenorhabditis elegans
6. Saccharomyces cerevisiae
7. Rattus norvegicus
8. Drosophila melanogaster
9. Bacillus subtilis
10. Pseudomonas aeruginosa PAO1
Prosite

- PROSITE consists of documentation entries describing protein domains, families and functional sites as well as associated patterns and profiles to identify them.
ENCODE / UCSC Genome Browser


ENCODE / UCSC Genome Browser

ENCODE / UCSC Genome Browser

- ENCODE: Encyclopedia of DNA Elements
- international collaboration of research groups
- funded by the National Human Genome Research Institute (NHGRI)
- build a comprehensive parts list of functional element in the human genome
- includes elements that act on protein and RNA level and regulatory elements
- The ENCODE Project Consortium. An Integrated Encyclopedia of DNA Elements in the Human Genome
taken from https://www.encodeproject.org/
Genomic Annotations

ENCODE Encyclopedia Overview

Top Level
- variant annotation
- chromatin states
- target genes of enhancers
- allele-specific events

Middle Level
- promoter-like
- enhancer-like
- transcript expression
- insulator-like/silencer-like

Ground Level
- DNase-seq (peaks)
- Hi-C (links, TADs, compartments)
- ChIA-PET (links)
- RBP (peaks, motifs, target genes)
- gene expression
- transcription start sites
- TF ChIP-seq (peaks, motifs, motif sites)
- histone mark ChIP-seq (peaks, domains)

available | under development | future plan

taken from https://www.encodeproject.org/data/annotations
UCSC Genome Browser

- actually a collection of integrated services
- [https://genome.ucsc.edu/index.html](https://genome.ucsc.edu/index.html)
- provides a more graphical interface to access the ENCODE data and a lot of additional tools
taken from https://genome-euro.ucsc.edu/cgi-bin/hgTracks?db=hg38&...
Databases - SQL

- Overlap with database lecture
- “SQL crash course”
- no design theory
- no normalization
- standard books like:

BioinfRes SoSe 17
More Books


BioinfRes SoSe 17
Selected SQL Topics

- Table modifications
  - insert, update, create, alter
- Data retrieval and reporting/aggregation
  - select, average, sum
- Combination and Performance
  - join
- Access control and permissions
  - grant
- Backup and Restore / Input-output
Reasons for DBMS

- redundancy, consistency
- limited access
- difficult multi-user access
- loss of information
- loss of integrity
- security issues
- expensive application development
Abstraction layers

View 1

Logical Layer

View 2

Physical Layer
Various Data Models

- Network model
- Hierarchical model
- **Relational Model**
- XML schema
- Object-oriented model
- Deductive model
Select Name
From Students, Attends, Lectures
Where Students.Matric = Attends.Matric and
    Attends.LectureNo = Lectures.LectureNo and
    Lectures.Title = 'Genomics';

Update Lectures
Set Title = 'Genomics of Mammalian'
Where LectureNo = 5;
Entity Relationship Model

- Graphical Notation
- Models real world “entities” and “relation”
- allows for “attributes”
- allows for functionalities (1:1, 1:n, n:m)
- allows to define keys
- key: a set for attributes which values combination allow unambiguous instance identification

BioinfRes SoSe 17
Notation

- **Student**: (strong) Entity
- **Name**: Attribute, key: underlined
- **Attends**: Relation
- **weak Entity (depend on others)**
Functionality

- **Student**
- **Attends**
- **Lecture**
- **Grade**

N: Many
M: One
Funktionalitäten

Studenten
- MatrNr
- Name
- Semester

Vorlesungen
- VorNr
- SWS
- Titel

Note
- prüfen

Assistenten
- PersNr
- Name
- Fachgebiet

Professoren
- PersNr
- Name
- Rang
- Raum

1
- arbeitenFür

N
- hören

M
- voraussetzen
- Nachfolger

N
- Vorgänger

taken from Prof. Kempers database lecture WS 13/14
BioinfRes SoSe 17
Prüfungen als schwacher Entitytyp

Studenten

ablegen

Prüfungen

MatrNr

N

umfassen

VorlNr

Vorlesungen

M

N

abhalten

PersNr

Professoren

Note

PrüfTeil

• Mehrere Prüfer in einer Prüfung
• Mehrere Vorlesungen werden in einer Prüfung abgefragt

taken from Prof. Kempers database lecture WS 13/14
SQL

- implemented by most available database management system manufacturer
- but: not always all specified features implemented
- not everything is specified!
- especially admin/server maintenance is often vendor specific
SQL Data Types

- char
- varchar
- binary and varbinary
- blob and text
- numeric, decimal, integer (exact)
- approximate: float, double
SQL Data Types

- various formats for time and date
- enum: one out of a defined set
- set: zero or more items out of a predefined list

For more information see the live tour through
ACID-Principle for Transactions

- **A**: Atomicity: All-or-nothing, i.e. a sequence of operations is executed like a single atomic operation which cannot be interrupted

- **C**: Consistency: After every operation the database is consistent, i.e. all conditions and constraints about context and relationships are fulfilled
ACID-Principle II

- I: Isolation: Concurrent operations to not affect each other
- D: Durability: Upon successful completion of a transaction it is guaranteed that all modifications are persistent, i.e. they are stored in the database, even in case of an unexpected power loss.
Relational Algebra

- $\sigma$ Selection
- $\pi$ Projection
- $\rho$ Rename
- $\times$ Cross Product
- $\Join$ Join
- $-$ Difference
- $\div$ Division
Relational Algebra

- ∪ Union
- ∩ Intersection
- ⋈ Semi Join (left)
- ⋈ Left Outer Join
- ⋈(Full) Outer Join
## Demonstration Table

<table>
<thead>
<tr>
<th>gene</th>
<th>indiv</th>
<th>organism</th>
<th>function</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>cytox</td>
<td>1</td>
<td>mouse</td>
<td></td>
<td>prep</td>
</tr>
<tr>
<td>gapdh</td>
<td>1</td>
<td>human</td>
<td>glycolysis</td>
<td>completed</td>
</tr>
<tr>
<td>gapdh</td>
<td>2</td>
<td>human</td>
<td>glycolysis</td>
<td>completed</td>
</tr>
<tr>
<td>ttn</td>
<td>2</td>
<td>human</td>
<td>muscle</td>
<td>ongoing</td>
</tr>
<tr>
<td>unkn0</td>
<td>3</td>
<td>human</td>
<td>NULL</td>
<td>prep</td>
</tr>
</tbody>
</table>
Selection

- The **SELECT** operation (denoted by $\sigma$ (sigma)) is used to select a subset of the tuples from a relation based on a selection condition.
- It acts as a (row) filter.
- Specified in the **WHERE**-clause.
- $\sigma$ status = “ongoing” (STATUS)
Selection

- General: the *select* operation is denoted by 
  \( \sigma_{<\text{selection condition}>}(R) \) where:
  - the \( \sigma \) (sigma) is used to denote the select operator
  - the selection condition is a Boolean (conditional) expression specified on the attributes of relation R
  - tuples that make the condition true are selected (appear in the result of the operation)
  - tuples that make the condition false are filtered out (discarded from the result of the operation)
Selection

- The Boolean expression specified in <selection condition> is made up of a number of clauses of the form:
  <attribute name> <comparison op> <constant value>
  or
  <attribute name> <comparison op> <attribute name>

- <attribute name> is the name of an attribute of R, <comparison op> id normally one of the operations \{=,>,>=,<,<=,!=\}

- Clauses can be arbitrarily connected by the Boolean operators **and**, **or** and **not**
Selection

- **NULL** is tested for with special operators
- Select $\sigma$ is commutative
- can be cascade of *select* operations of a conjunction of conditions:
  $\sigma_{<\text{condition}_1>}(\sigma_{<\text{condition}_2>}(R)) = \sigma_{<\text{condition}_2>}(\sigma_{<\text{condition}_1>}(R))$

  $\sigma_{<\text{cond}_1>}(\sigma_{<\text{cond}_2>}(\sigma_{<\text{cond}_3>}(R))) = \sigma_{<\text{cond}_1> \text{ AND } <\text{cond}_2> \text{ AND } <\text{cond}_3>}(R)$
Projection

- **PROJECT** Operation is denoted by $\pi$ (pi)
- use PROJECT to retrieve specific attributes of relation R
- It acts as a (column) filter of the tuples
- Example:
  $$\pi_{\text{Gene, status}} (\text{STATUS})$$
- Project removes duplicates which might occur (in SQL: SELECT DISTINCT instead of simple SELECT)
Single Expression vs. Sequence of Relational Operations

To retrieve completed genes from our example:

- Single expression:
  \[ \pi_{\text{gene, status}}(\sigma_{\text{status=completed}}(\text{STATUS})) \]

- Sequence of operation:
  \[
  \text{ALL}_{-}\text{COMP} \leftarrow \sigma_{\text{status=completed}}(\text{STATUS}) \\
  \text{RESULT} \leftarrow \pi_{\text{gene, status}}(\text{ALL}_{-}\text{COMP})
  \]
Rename

- \textit{RENAME} is denoted by $\rho$ (rho)
- In some cases, we may want to rename the attributes of a relation or the relation name or both
  - Useful when a query requires multiple operations
  - Necessary in some cases (see JOIN operation later)
RENAME

- RENAME operations $\rho$ can be expressed by any of the following forms:
  - $\rho_S(R)$ changes: the *relation name* only to $S$
  - $\rho_{(B_1, B_2, \ldots, B_n)}(R)$ changes: the *column (attribute)* names only to $B_1, B_1, \ldots, B_n$
  - $\rho_{S(B_1, B_2, \ldots, B_n)}(R)$ changes both: the relation name to $S$, *and* the column (attribute) names to $B_1, B_1, \ldots, B_n$
Relational Operators from Set Theory

- Union
- Intersection
- Minus
- Cartesian Products
Union

- It is a Binary operation, denoted by $\cup$
- The result of $R \cup S$, is a relation that includes all tuples that are either in $R$ or in $S$ or in both $R$ and $S$
- Duplicate tuples are eliminated
- $R$ and $S$ have to type compatible:
  - they must have the same number of attributes
  - corresponding attributes are type compatible
Intersection

- INTERSECTION is denoted by $\cap$
- The result of the operation $R \cap S$, is a relation that includes all tuples that are in both $R$ and $S$
- The attribute names in the result will be the same as the attribute names in $R$
- The two operand relations $R$ and $S$ must be “type compatible”
Set Difference

- SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by $-$
- The result of $R - S$, is a relation that includes all tuples that are in $R$ but not in $S$
- The attribute names in the result will be the same as the attribute names in $R$
- The two operand relations $R$ and $S$ must be “type compatible”
Properties of Union, Intersection and Difference

- Both union and intersection are commutative; that is:  
  \[ R \cup S = S \cup R, \text{ and } R \cap S = S \cap R \]

- Union and intersection are associative operations; that is:  
  \[ R \cup (S \cup T) = (R \cup S) \cup T \]
  \[ (R \cap S) \cap T = R \cap (S \cap T) \]

- The minus operation is not commutative; that is:  
  \[ R - S \neq S - R \]
Cross Product (Cartesian Product)

- CROSS PRODUCT Operation
- Used to combine tuples from two relations in a combinatorial fashion
- Denoted by $R(A_1, A_2, \ldots, A_n) \times S(B_1, B_2, \ldots, B_m)$
- Result is a relation $Q$ with degree $n + m$ attributes:
  $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$
Cartesian Product (Cross Product)

- The resulting relation contains every possible combination of the tuples from R and S -- one from R and one from S

- Hence, if R has $n_R$ tuples (denoted as $|R| = n_R$), and S has $n_S$ tuples, then $R \times S$ will have $n_R \times n_S$ tuples

- The two operands do NOT have to be "type compatible"

- Generally, CARTESIAN PRODUCT is not a meaningful operation, but can become meaningful when followed by other operations
Join

- JOIN Operation (denoted by $\bowtie$)
- Sequence of CARTESIAN PRODUCT followed by SELECT is used to identify and select related tuples from two relations
- very important for any relational database with more than a single relation, because it allows to combine related tuples from various relations
Join

- The general form of a join operation on two relations $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$ is:
  \[
  R \bowtie_{\text{join condition}} S
  \]
- where $R$ and $S$ can be any relations that result from general relational algebra expressions
Join

Consider the following JOIN operation:

- If $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$
  
  Think about $R.A_i = S.B_j$

- Result is a relation $Q$ with degree $n + m$ attributes:
  
  $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$

- The resulting relation state has one tuple for each combination of tuples – $r$ from $R$ and $s$ from $S$, but only if they satisfy the join condition $r[A_i]=s[B_j]$

- if $R$ has $n_R$ tuples, and $S$ has $n_S$ tuples, then the join result will generally have less than $n_R \times n_S$ tuples
Join (more precise)

- The general case of JOIN operation is called a Theta-join: \( R \bowtie_{\theta} S \)
- The join condition is called theta
- Theta can be any general boolean expression on the attributes of R and S; for example: \( R.A_i < S.B_j \) AND (\( R.A_k = S.B_l \) OR \( R.A_p < S.B_q \))
Equijoin

- The most common use of join involves join conditions with equality comparisons only
- Such a join, where only the comparison operator used is =, is called an EQUIJOIN
- The JOIN seen in the previous example was an EQUIJOIN
Natural Join

- Another variation of JOIN called NATURAL JOIN — denoted by * or □ without condition.
- It was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition.
- \[ Q \leftarrow R(A,B,C,D) \ast S(C,D,E) \]
- implicit join condition includes each pair of attributes with the same name, “AND”ed together:
  \[ R.C=S.C \text{ AND } R.D = S.D \]
- keeps only one attribute of each such pair:
  \[ Q(A,B,C,D,E) \]
Semi Join

- acts like a filter based on a specified attribute
- $R \Join S$ means: if $R$ and $S$ have a common attribute $C$ the result are all tuples from $R$ which $C$ value occurs also in $S$, $n_Q \leq n_R$ tuples
- $Q \leftarrow R(A,B,C) \Join S(C,D,E)$
- $Q(A,B,C)$ with $n_R$ attributes
- $\pi_{A,B,C} (\sigma_{R.C=S.C}(R \times S))$
Left Outer Join

- Right version is analogous
- add information to corresponding left side tuples
- $R \bowtie S$ means: if $R$ and $S$ have a common attribute $C$ the result are all combined tuples from $R$ and $S$ where $R.C = S.C$ and in addition all remaining tuples from $R$, $n_Q = n_R$ tuples
- $Q \leftarrow R(A,B,C) \bowtie S (C,D,E)$
- $Q(A,B,C,D,E)$ with $n_{RUS}$ attributes
- if no matching tuples found in $S$ attributes $D$ and $E$ contain no values
(Full) Outer Join

- combines corresponding tuples von R and S where possible, else attributes left blank

- \(R \bowtie S\) means: if R and S have a common attribute C the result are all combined tuples from R and S where \(R.C = S.C\) and in addition all remaining tuples from R and S, \(n_Q \leq n_{R+S}\) tuples

- \(Q <- R(A,B,C) \bowtie S (C,D,E)\)

- \(Q(A,B,C,D,E)\) with \(n_{R∪S}\) attributes

- if no matching tuples found in R or S attributes A, B or D and E contain no values
Division

- Gives all attribute tuple for R-S where a values for R-S co-occurs with all tuples in S
- R(A,B) and S(B)
- R÷S: Q(A) where each result tuple in Q can be found in R in combination with every tuple from S
Complete Set of Relational Operations

- The set of operations including SELECT $\sigma$, PROJECT $\pi$, UNION $\cup$, DIFFERENCE $-$, RENAME $\rho$, and CARTESIAN PRODUCT $\times$ is called a complete set because any other relational algebra expression can be expressed by a combination of these five operations.

- Examples:
  - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$
  - $R \bowtie_{<\text{join condition}>} S = \sigma_{<\text{join condition}>} (R \times S)$
Beyond Classical Algebra

- Grouping: group by
- Aggregation: count, sum, average, min, max
Keys and Indexes

- Each relation represents a subset of the cartesian product of its domains (attributes)
- Some values might be unique for a row others are not
- To address and access a specific tuple in a relation we need to define a primary key
- A primary key is set of attributes which combination allows us to unambiguously identify a certain row in the relation
Keys and Indexes

- Consequences:
  - Each primary key (combination) can occur only once in a table
  - Entries which miss at one of these attribute values are not allows (NOT NULL)
  - Default values for these attributes make no sense
  - These system has to keep track which the help of an index

- The key depends on the modeling and the domain
Indexes/Constraints

- PRIMARY KEY: UNIQUE, NOT NULL
- UNIQUE: If there is a value it must be unique, if there is no value but NULL it can occur multiple times
- INDEX: A search structure which allows to find tuples (rows) which a specific attribute value efficiently
  - must explicitly requested in the table structure
  - for character types you can the prefix length
Performance Considerations

- There are three relations to join A*B*C:
  - A (1,000,000 rows)
  - B (100 rows)
  - C (10,000 rows)
Performance Considerations

- (Worst) Case w/o indexes and bad sequence:
  A*C: 10.000.000.000 comparisons O(n*m) ->
  D(10.000.000.000 rows)
  D*B(1.000.000.000.000 comparisons) O(n*m)
  - of course tuples might be dropped in reality because of missing join partners

- Case with indexes and clever sequence:
  B*A: 100* log(10.000.000) comparisons -> D (10.000.000 rows)
  C*D: 10.000 * log(10.000.000) comparisons
Performance Considerations

- Sequence of evaluation can be optimized by the database engine
  - clever order with exploitation of associativity and commutativity
  - example: $100 \times \log(10,000,000)$ vs $10,000,000 \times \log(100)$
  - maybe not effective in worst case but definitely everytime else