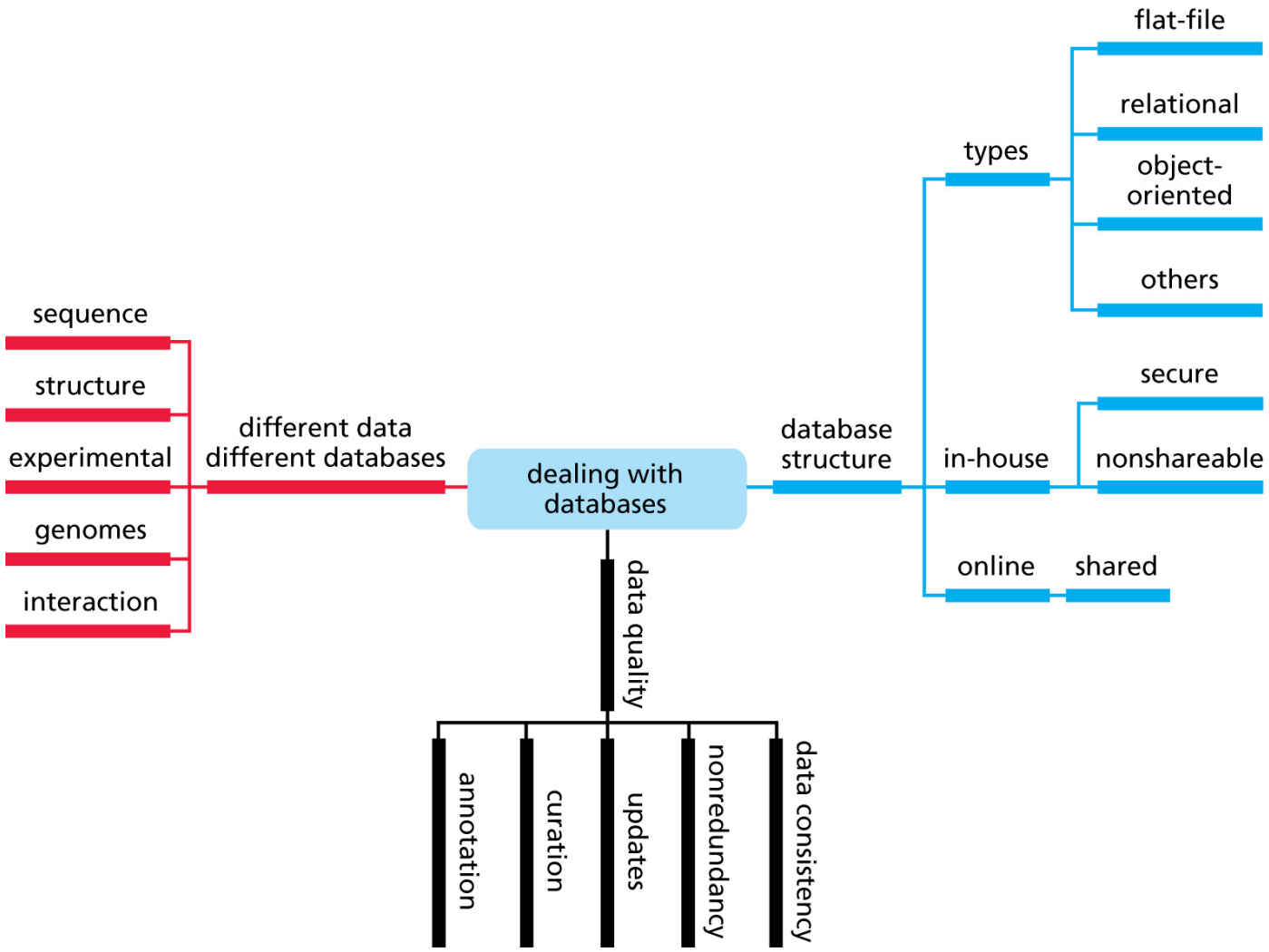
Biological Databases and Datamining

Lecture 1: Dealing with Databases and basic SQL



Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

Examples of flat-file database structures

(A)

NAME	TELEPHONE	ADDRESS
S. Claus	0203 450	The North Pole, Lapland
M. Mouse	0202 453	Disneyworld, Florida
A. Moonman	0104 459	Craterland, The Moon

(B) GenBank Flat-File Format

```
LOCUS       SCU49845     5028 bp    DNA
DEFINITION  Saccharomyces cerevisiae TCP1-beta gene, partial cds, and
             Axl2p
            (AXL2) and Rev7p (REV7) genes, complete cds.
ACCESSION   U49845
VERSION     U49845.1  GI:1293613
KEYWORDS    .
SOURCE      Saccharomyces cerevisiae (baker's yeast)
  ORGANISM  Saccharomyces cerevisiae
            Eukaryota; Fungi; Ascomycota; Saccharomycotina;
           Saccharomycetes;
            Saccharomycetales; Saccharomycetaceae; Saccharomyces.
```

Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

Example of tables in a relational database.

protab1

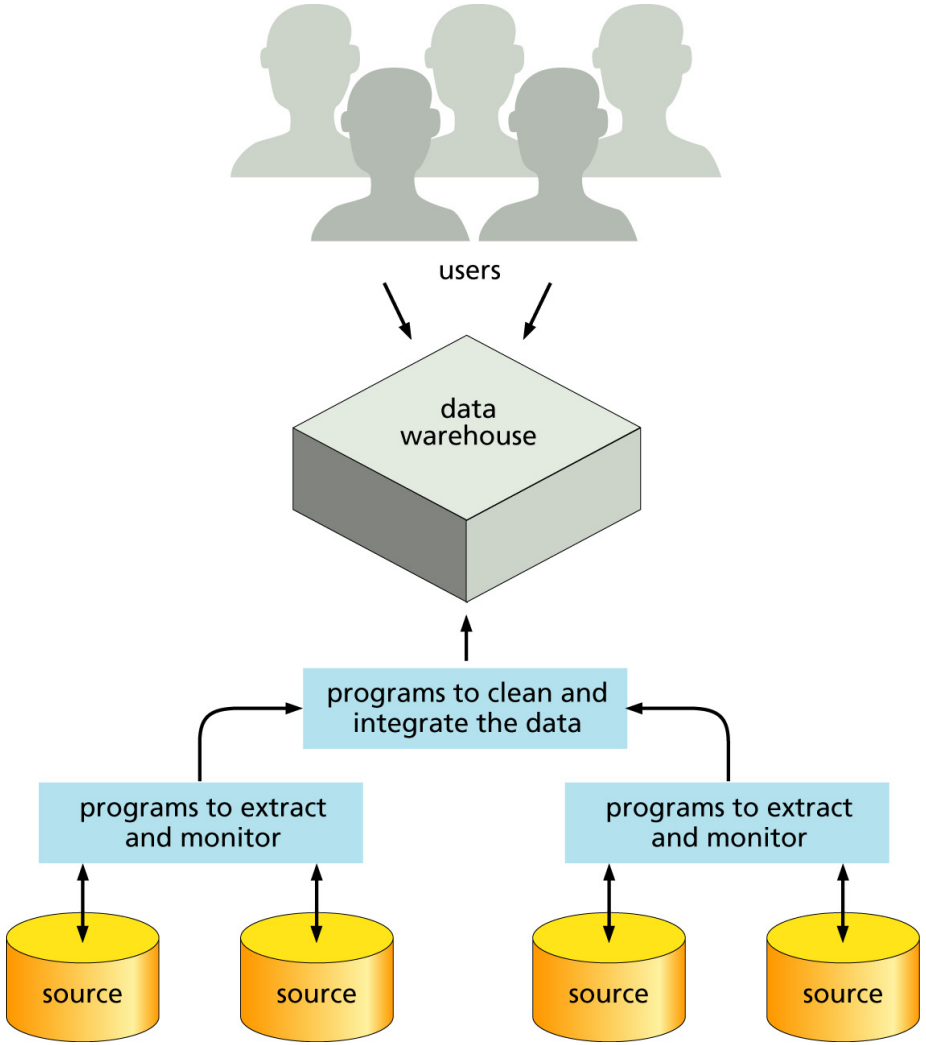
Protein-code	Protein-name	Length	Species-origin
P1001	Hemoglobin	145	Bovine
P1002	Hemoglobin	136	Ovine
P1003	Eye Lens Protein	234	Human
.....			

protab2

Protein-code	Protein-sequence
P1001	MDRTTHGFDLKLLSPRTVNQWLMLALFFGHS...
P1002	MDKTSHGFEIKLLTPKKLQQWLMIAIYFGHT...
P1003	SRTHEEEGKLMQWPPRPLYIALFTEPPYP...
.....	

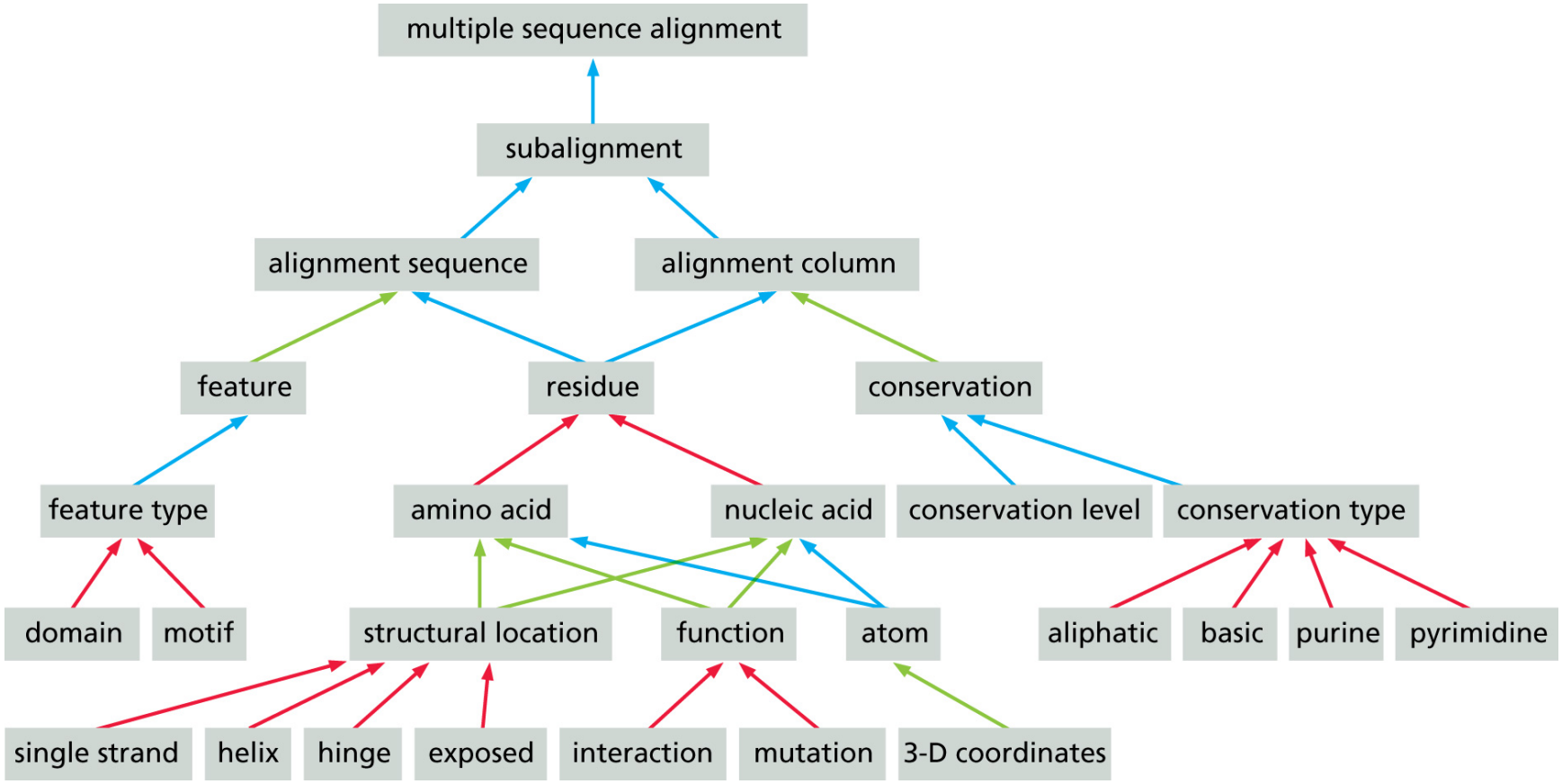
- Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

Data warehousing



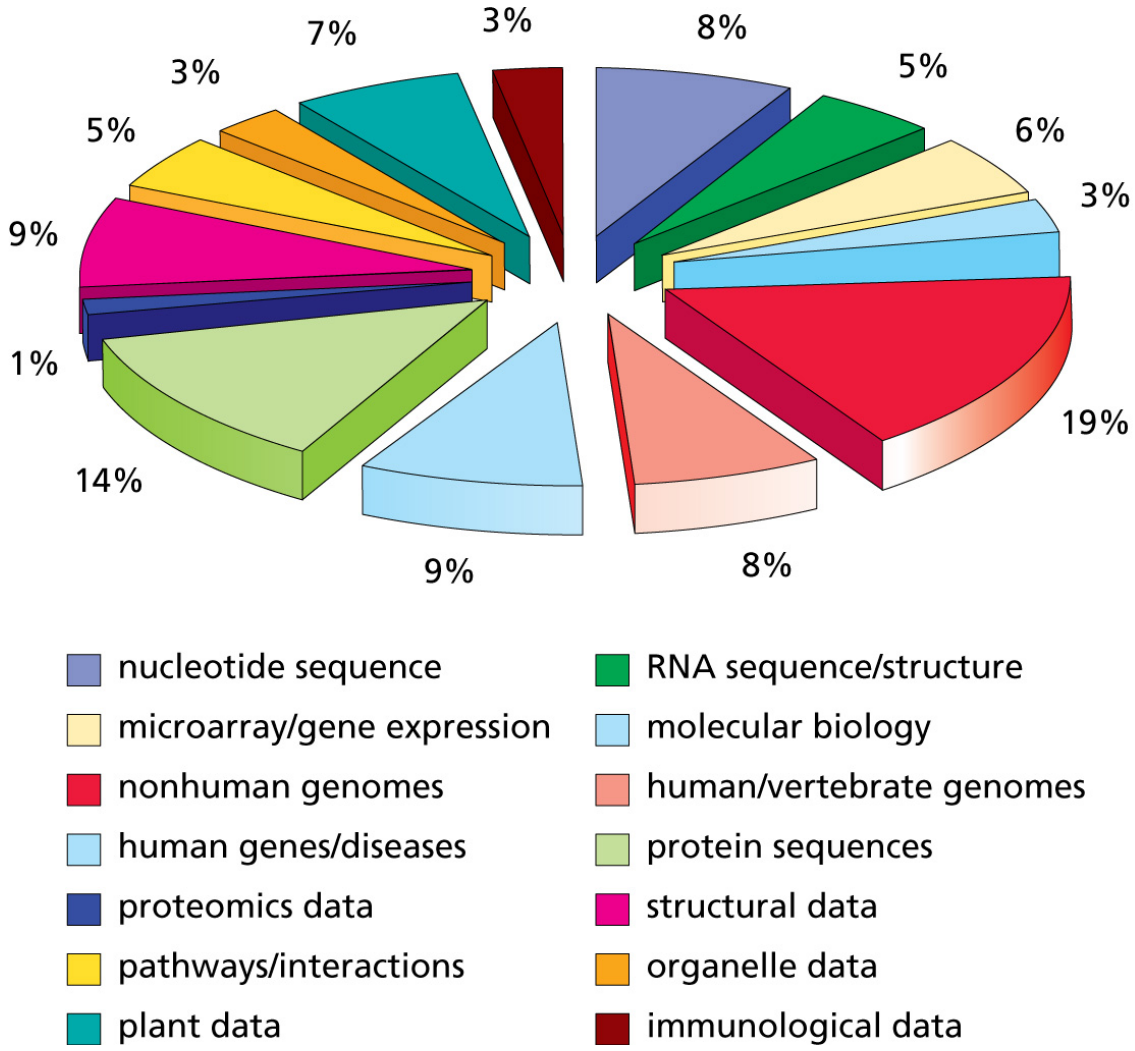
- Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

MAO: Multiple alignment ontology



- Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

Distribution of different types of databases as classified by NAR (Nucleic Acid Research)



- Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

Extract from a GenBank DNA sequence file

```
LOCUS       NM_005358               7235 bp    mRNA    linear   PRI 02-AUG-2006
DEFINITION  Homo sapiens LIM domain 7 (LMO7), mRNA.
ACCESSION   NM_005358
VERSION     NM_005358.4  GI:111119012
KEYWORDS    .
SOURCE      Homo sapiens (human)
…
gene            1..7235
                /gene="LMO7"
                /note="synonyms: LOMP, FBX20, FBXO20, KIAA0858"
                /db_xref="GeneID:4008"
                /db_xref="HGNC:6646"
                /db_xref="HPRD:05078"
                /db_xref="MIM:604362"
CDS             1261..5310
                /gene="LMO7"

ORIGIN
```

```
       1 ggaaagaagt ggaataatta ggaacctagg gtggggtagg gtagcaggac atttcaaaca
      61 ttaatgagca tatgagattc caggtcttgt taaaatgcaa attctgattc agctggtagg
     121 tgaggtctga gattgtgcat ttctaacaag cactcagata atcttaaggc tgttggcccc
     181 agggtcacac ttatagtgat tttctagaac ccagttgggg aagtgaatct tgggcaggag
     241 aaatacacac ctcttgcatt gagtttggag atctcatctg atataacttt ttaagaaaga
     301 aaaataattt tccaaatatc caattgataa gctttcccac taagtggctt tccacactaag
     361 tggctgcgtt atgaaaattg cttcactttg aaacttctgg tcttggtaat atagaatttc
     421 tgtgttctca cagtgcttga ttgagaatat gatattgaga ttatggcata aaatatagtg
     481 gctgtacaaa aaaaaataca ttattaggat ctctaacaat tatgtaaaag tcattgcttc
     541 atgggtagag ctcaaacttt ggtgtgagac ctggttttat tcttggcact tactctgagt
     601 tgtcttaggc aaattaatac cttaagcaaa aatattctca tgtacatttt acatgagaat
     661 tataaatgaa gtacataaag tccagcagtc acaaatgtta tctattatta ccatcgtcct
     721 aagactgcaa tcagctatag tgaaagtagt ctcaaagatt gtttcataaa tcatcagatt
     781 cacctaattt tctaaagaat ttaaataagg agatggaatg aatagattgc attttgtttc
     841 catgcacagg ggaactgtgc atatttcttc tgtgactcgg aaatggttta actttttaaaa
     901 atcccaaaat agctgaagtt agcagacatg caatttacca aggatgattg gaatttttat
     961 ctttcctgta ataatactat acccaagcac actgctcatg aggaaaacat ttttatgtga
    1021 atcttttact cttgggggca aagaatgctg tttttctttt tgataactat gtttatagaa
    1081 tctaaatcac cctgagcaat tatttcaaca tctaaagtta ttattaccat tcatgtttca
    1141 tttatagcta tttgaatttt gatgaatttc aatatggtgc tacagtgata gggcaagtgc
    1201 aaataagttc aatatatggg tacggtctaa agctatttta atttttttat tacaactgct
    1261 atgaagaaaa ttaggatatg ccatattttc acgtttaca gttggatgtc ctatgatgtt
    1321 ctcttccaga gaacagagct cggagctctg gaaatttgga ggcaactgat atgtgctcat
    1381 gtctgcatct gtgtgggttg gctgtatctc agggacagag tctgcagcaa aaaagatata
    1441 attttgagga ctgaacaaaa ttcaggaagg actattctca ttaaggcagt aacagagaag
    1501 aattttgaaa caaaagattt tcgagcctct ctagaaaatg gtgttctgct gtgtgatttg
    1561 attaataagc ttaaacctgg cgtcattaag aagatcaata gactgtctac accaatagca
    1621 ggattggata atataaacgt tttcttgaaa gcttgtgaac agattggatt gaaagaagcc
    1681 cagcttttcc atcctggaga tctacaggat ttatcaaatc gagtcactgt caagcaagaa
    1741 gagactgaca ggagagtgaa aaatgttttg ataacattgt actggctggg aagaaaagca
    1801 caaagcaacc cgtactataa tggtccccat cttaatttga aagcgtttga gaatctttta
    1861 ggacaagcac tgacgaaggc actcgaagac tccagcttcc tgaaaagaag tggcagggac
    1921 agtggctacg gtgacatctg gtgtcctgaa cgtggagaat ttcttgctcc tccaaggcac
    1981 catagaggag aagattcctt tgaagcttg gactcttgg gctcgaggtc atgacaagcc
    2041 tgctcctctg atatcacgtt gagaggggcg cgtgaaggtt ttgaaagtga cacagattcg
    2101 gaatttacat ttaagatgca ggattataat aaagatgata tgtcgtatcg aaggatttcg
    2161 gctgttgagc caaagactgc gttacccttc aatcgttttt tacccaacaa aagtagacag
    2221 ccatcctatg taccagcacc tctgagaaag aaaaagccag acaaacatga ggataacaga
    2281 agaagttggg caagcccggt ttatacagaa gcagatggaa cattttcaag actctttcaa
    2341 aagatttatg gtgagaatgg gagtaagtcc atgagtgatg tcagcgcaga agatgttcaa
    2401 aacttgcgtc agctgcgtta cgaggagatg cagaaaataa aatcacaatt aaaagaacaa
    2461 gatcagaaat ggcaggatga ccttgcaaaa tggaaagatc gtcgaaaaag ttacacttca
    2521 gatctgcaga agaaaaaaga agagagagaa gaaattgaaa agcaggcact tgagaagtct
    2581 aagagaagct ctaagacgtt taaggaaatg ctgcaggaca gggaatccca aaatcaaaag
    2641 tctacagttc cgtcaagaag gagaatgtat tcttttgatg atgtgctgga ggaaggaaag
    2701 cgacccccta caatgactgt gtcagaagca agttaccaga gtgagagagt agaagagaag
    2761 ggagcaactt atccttcaga aattcccaaa gaagattcta ccacttttgc aaaaagagag
    2821 gacgtgtaa caactgaaat tcagcttcct tctcaaagtc ctgtggaaga acaaagccca
    2881 gcttcttgt cttctctgcg ttcacggagc acacaaatgg aatcaactcg tgtttcagct
    2941 tctctcccca gaagttaccg gaaaactgat acagtcaggt taacatctgt ggtcacacca
    3001 agacccttg gctctcagac aaggggaatc tcatcactcc ccagatctta cacgatggat
    3061 gatgcttgga agtataatgg agatgttgaa gacattaaga gaactccaaa caatgtggtc
    3121 agcacccctg caccaagccc ggacgcaagc caactggctt caagcttatc tagccagaaa
    3181 gaggtagcag caacagaaga agatgtgaca aggctgccct ctcctacatc ccccttctca
    3241 tctctttccc aagaccaggc tgccacttct aaagccacat tgtcttccac atctggtctt
    3301 gatttaatgt ctgaatctgg agaaggggaa atctccccac aaagagaagt ctcaagatcc
    3361 caggatcagt tcagtgatat gagaatcagc ataaaccaga cgcctgggaa gagtcttgac
    3421 tttgggttta caataaaatg ggatattcct gggatcttcg tagcatcagt tgaagcaggt
    3481 agcccagcag aattttctca gctacaagta gatgatgaaa ttattgctat taacaacacc
    3541 aagttttcat ataacgattc aaaagagtgg gaggaagcca tggctaaggc tcaagaaact
    3601 ggcacctag tgatggatgt gaggcgctat ggaaaggctg gttcacctga aacaaagtgg
    3661 attgatgcaa cttctggaat ttacaactca gaaaaatctt caaatctatc tgtaacaact
    3721 gatttctccg aaagccttca gagttctaat attgaatcca aagaaatcaa tggaattcat
    3781 gatgaaagca atgcttttga atcaaaagca tctgaatcca tttctttgaa aaacttaaaa
    3841 aggcgatcac aatttttttga acaaggaagc tctgattcgg tggttcctga tcttccagtt
    3901 ccaaccatca gtgcccccag tcgctgggtg tgggatcaag aggaggagcg gaagcggcag
    3961 gagaggtggc agaaggagca ggaccgccta ctgcaggaaa aatatcaacg tgagcaggag
    4021 aaactgaggg aagagtggca aagggccaaa caggaggcag agagagagaa ttccaagtac
    4081 ttggatgagg aactgatggt cctaagctca aacagcatgt ctctgaccac acgggagccc
    4141 tctcttgcca cctgggaagc tacctggagt gaagggtcca agtcttcaga cagagaagga
    4201 acccgagcag gagaagagga gaggagacag ccacaagagg aagttgttca tgaggaccaa
    4261 ggaaagaagc cgcaggatca gcttgttatt gagagagaga ggaaatggga gcaacagctt
    4321 caggaagagc aagagcaaaa gcggcttcag gctgaggctg aggagcagaa gcgtcctgcg
    4381 gaggagcaga agcgccaggc agagatagag cgggaaacat cagtcagaat ataccagtac
    4441 aggaggcctg ttgattccta tgatatacca aagacagaag aagcatcttc aggttttctt
    4501 cctggtgaca ggaataaatc cagatctact actgaactgg atgattactc cacaaataaa
    4561 aatggaaaca ataaatattt agaccaaatt gggaacatga cctcttcaca gaggagatcc
    4621 aagaaagaac aagtaccatc aggagcagaa ttggagaggc aacaaatcct tcaggaaatg
    4681 aggaagagaa caccccttca caatgacaac agctggatcc gacagcgcag tgccagtgtc
    4741 aacaaagagc ctgttagtct tcctgggatc atgagaagag gcgaatcttt agataacctg
    4801 gactcccccc gatccaattc ttggagacag cctccttggc tcaatcagcc cacaggattc
    4861 tatgcttctt cctctgtgca agacttttagt cgcccaccac ctcagctggt gtccacatca
    4921 aaccgtgcct acatgcggaa ccccctcctcc agcgtgcccc caccttcagc tggctccgtg
    4981 aagacctcca ccacaggtgt ggccaccaca cagtccccca ccccgagaag ccattccact
    5041 tcagcttcac agtcaggctc tcagctgcgt aacaggtcag tcagtgggaa gcgcatatgc
    5101 tcctactgca ataacattct gggcaaagga gccgccatga tcatcgagtc cctgggtctt
    5161 tgttatcatt tgcattgttt taagtgtgtt gcctgtgagt gtgacctcgg aggctctttcc
    5221 tcaggagctg aagtcaggat cagaaaccac caactgtact gcaacgactg ctatctcaga
    5281 tccaaatctg gacggccaac cgccatgtga tgtaagcctc catacgaaag cactgttgca
    5341 gatagaagaa gaggtggttg ctgctcatgt agatctataa atatgtgttg tatgtctttt
    5401 ttgcttttct tttaaaaaaa agaataactt tttttgcctc tttagattac atagaagcat
    5461 tgtagtcttg gtagaaccag tatttttgtt gtttatttat aaggtaattg tgtgtgggga
    5521 aaagtgcagt atttacctgt tgaattcagc atcttgagag cacaaggaaa aaaataagaa
    5581 cctacgaata tttttgaggc agataatgat ctagtttgac tttctagtta gtggtgtttt
    5641 gaagaggtta tttattgtt tttaaaaaa aggttcttaa acatttctg aaatagtaaa
    5701 tataaataca taattgcatt tgctctgttt attgtaatgt attctaaatt aatgcagaac
    5761 catatggaaa atttcattaa aatctatccc caaatgtgct ttctgtatcc ttccttctac
    5821 ctattattct gatttttaaa aatgcagtta atgtaccatt tatttgcttg atgaagggag
    5881 ctctattttc tttaccagaa atgttgctaa gtaattccca atagaaagct gcttattttc
    5941 attaatgaaa aataaccatg gtttgtatac tagaagtctt cttcagaaac tggtgagcct
    6001 ttctgttcaa ttgcatttgt aaataaactt gctgatgcat ttaacgagtg ggtcgtcttt
    6061 ttcttaggtg tatgtgtctg acctcaggcc ttttagccat atttcagtat gtggccttt
    6121 ttgatgttat gttttatcca gtagctttac taaggtataa ttgatgtaat aaactgcata
    6181 tatttaaagt gtatactttg acaaattttg acatggtgta taccttcgaa actatgccac
    6241 agtctggatg tgtttactga aacattttaa taaggaagtt tatttttgat aaagttatgt
    6301 ttttggatac aatatatttg tatggtgaga gtgatgaatt gttggatcat ttgatctttt
    6361 actaccccca tgataaaagg agaagacaac agtgagctta gaatatctat aaagcaaaaa
    6421 atgtagtctc ttgtttaaaa aatctggagc gggaatgcaa ggatacaaaa ctttagcatg
    6481 ctttggagcaa aaatttaaac ttactggaat cttttataat aatgtaagtg gaatggagga
    6541 ttctaggaac tgagaactgt attggaatag gttcaaaata tgtaagaaat gctaatgtgg
    6601 gagataaaaa ttttatttag tacttattct gattattatt aaagtaataa tgtgttcctt
    6661 gaggataact tgtcaaatgc cccaaagcat aaagaatata attctgaatc ccaaattcca
    6721 aagacaagaa ctctgtgttt gaattcattc tgcatataat tatttataag tatagattgt
    6781 gaatttttcc atgttcttaa aattattttt atcttttttc atggttgcat agtgctccat
    6841 tgtttggcct tggtaatatt tagttgataa ttccattact gtgtatttct cacttgtttc
    6901 taagatcaaa catttttaata tgtgcatgtt atatataaat atgtaaattc tgtgatactc
    6961 tatgatcatc tctttctttta tattattttc atagacatga aatagttgct cagagattat
    7021 gcatttttaag acactcatag tatatattgc caaagtggtt tccagaaagg cactgctggc
    7081 ttcgactcct ataagcagca cgtgggcttg ttcatctcac tgcatgttta tgaagataca
    7141 gttcttttgc cttgttctct gcctgatgtg tatgcagagg cagccctcaa tatgcagtgg
    7201 ttgaataaat gaatgaagaa accactaaaa aaaaa
```

- Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

LIM domain 7 entry in Swiss-Prot database

Note: most headings are clickable, even if they don't appear as links. They link to the user manual or other documents.

Entry information

Entry name	**LMO7_HUMAN**
Primary accession number	**Q8WWI1**
Secondary accession numbers	O15462 O95346 Q9UKC1 Q9UQM5 Q9Y6A7
Integrated into Swiss-Prot on	March 15, 2004
Sequence was last modified on	March 15, 2004 (Sequence version 2)
Annotations were last modified on	July 25, 2006 (Entry version 39)

Name and origin of the protein

Protein name	**LIM domain only protein 7**
Synonyms	**LOMP** **F-box only protein 20**
Gene name	**Name:** **LMO7** Synonyms: FBX20, FBXO20, KIAA0858
From	Homo sapiens (Human) [TaxID: 9606]
Taxonomy	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo.

References

[1] NUCLEOTIDE SEQUENCE [MRNA] (ISOFORM 3), AND TISSUE SPECIFICITY.
TISSUE=Brain, and Peripheral blood leukocyte;
DOI=10.1007/s00439-001-0646-6; PubMed=11935316 [NCBI, ExPASy, EBI, Israel, Japan]
Rozenblum E., Vahteristo P., Sandberg T., Bergthorsson J.T., Syrjakoski K., Weaver D., Haraldsson K., Johannsdottir H.K., Vehmanen P., Nigam S., Golberger N., Robbins C., Pak E., Dutra A., Gillander E., Stephan D.A., Bailey-Wilson J., Juo S.-H.H., Kainu T.,  , Kallioniemi O.-P.;
"A genomic map of a 6-Mb region at 13q21-q22 implicated in cancer development: identification and characterization of candidate genes.";
Hum. Genet. 110:111-121(2002).

Key	From	To	Length	Description	FTId
CHAIN	1	1683	1683	LIM domain only protein 7.	PRO_0000075824
DOMAIN	54	168	115	CH.	
DOMAIN	1042	1128	87	PDZ.	
DOMAIN	1612	1678	67	LIM zinc-binding.	

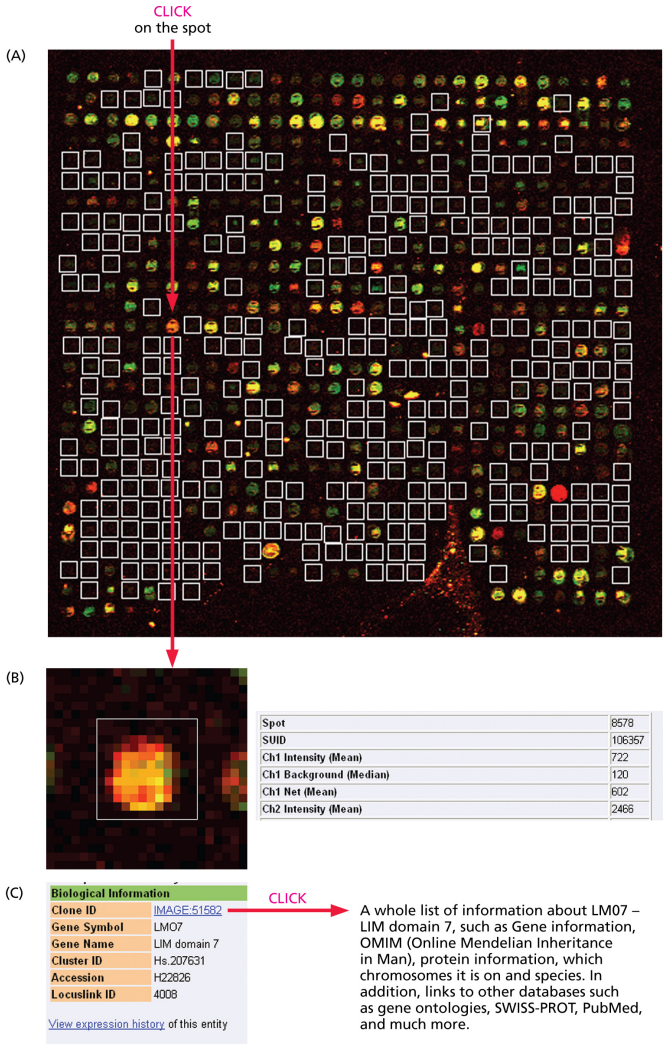
```
        10         20         30         40         50         60
MKKIRICHIF TFYSWMSYDV LFQRTELGAL EIWRQLICAH VCICVGWLYL RDRVCSKKDI

        70         80         90        100        110        120
ILRTEQNSGR TILIKAVTEK NFETKDFRAS LENGVLLCDL INKLKPGVIK KINRLSTPIA

       130        140        150        160        170        180
GLDNINVFLK ACEQIGLKEA QLFHPGDLQD LSNRVTVKQE ETDRRVKNVL ITLYWLGRKA
```

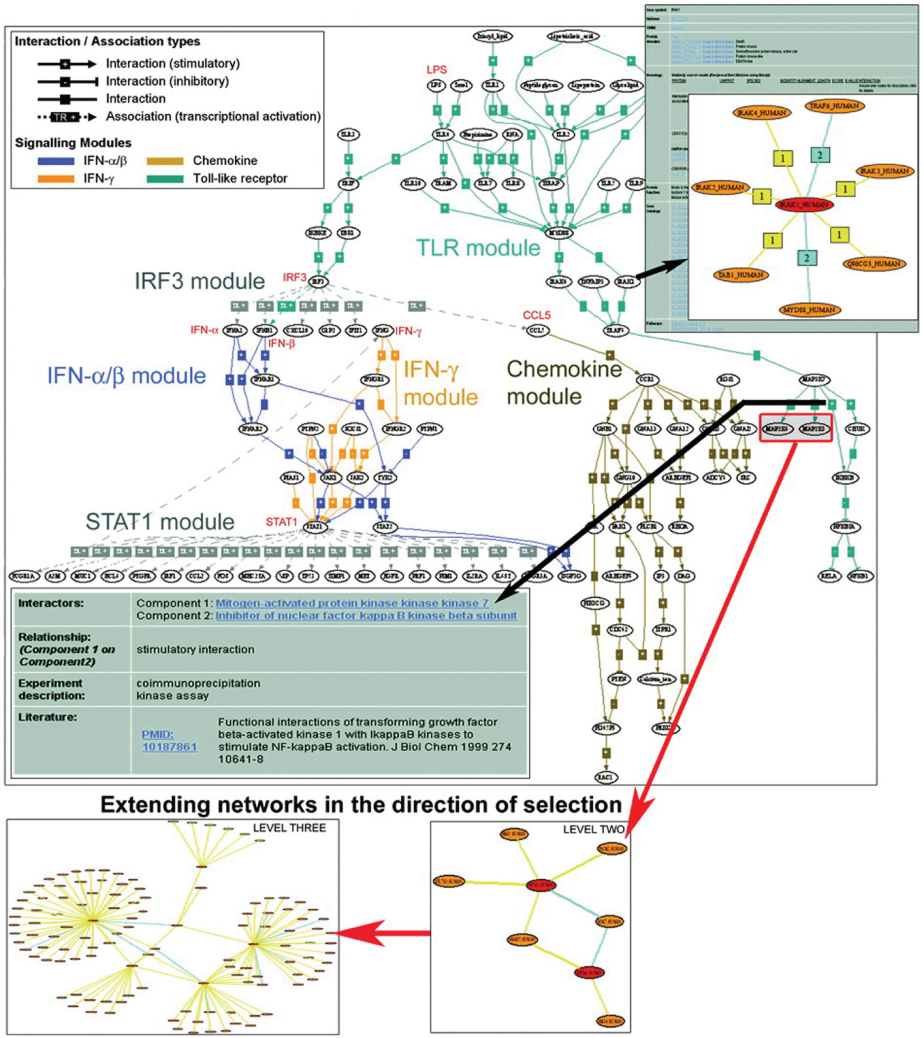
- Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

Stanford Microarray database



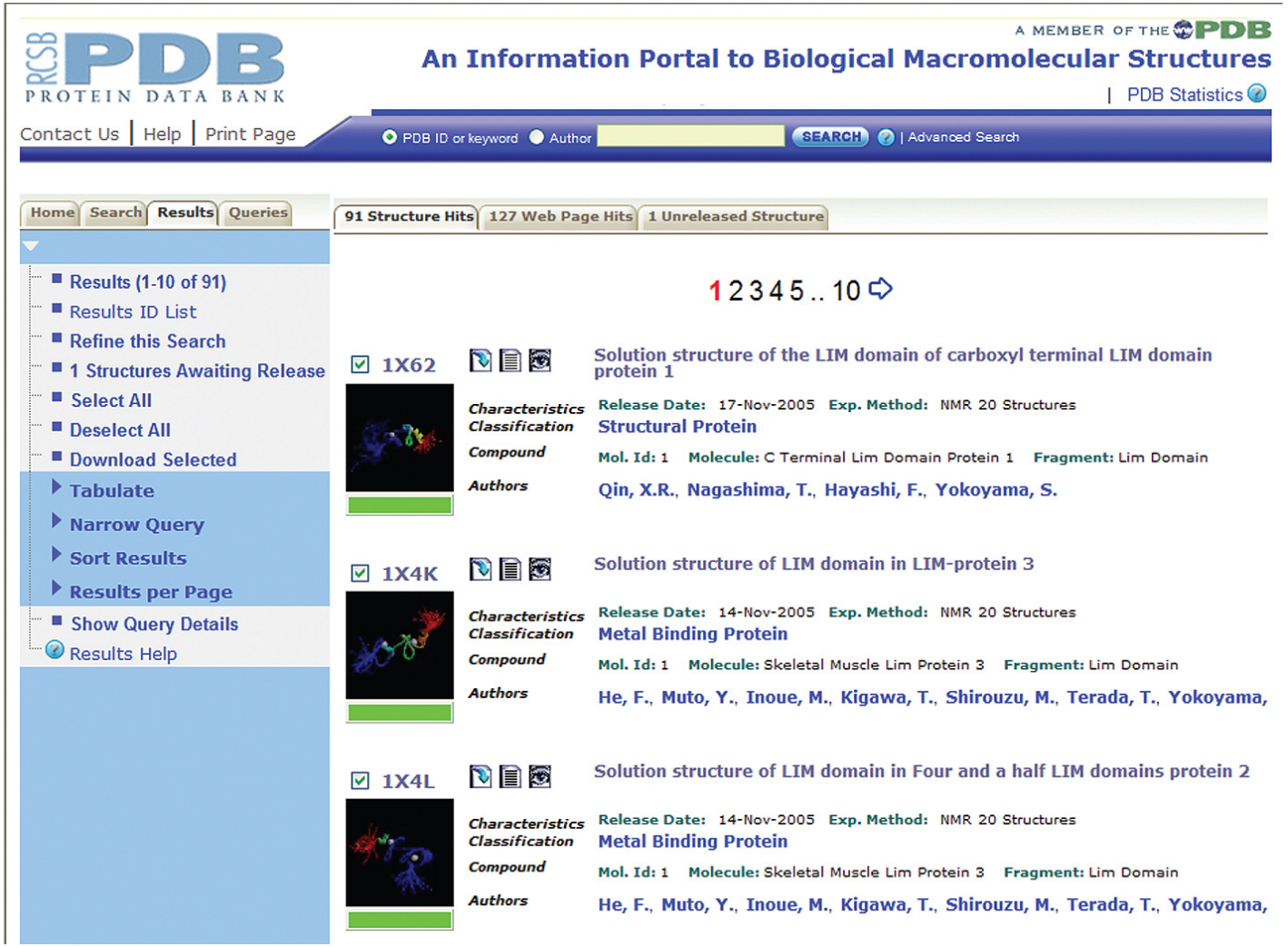
- Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

Interaction map generated by pSTIING



- Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

Protein Structure databases



- Zvelebil and Baum, Understanding Bioinformatics, Garland Science (2008)

Data Quality

- Ways to identify inaccuracies
  - Computer-based
  - Manual curation
  - Non-redundancy
  - Data consistency
  - Updating the database

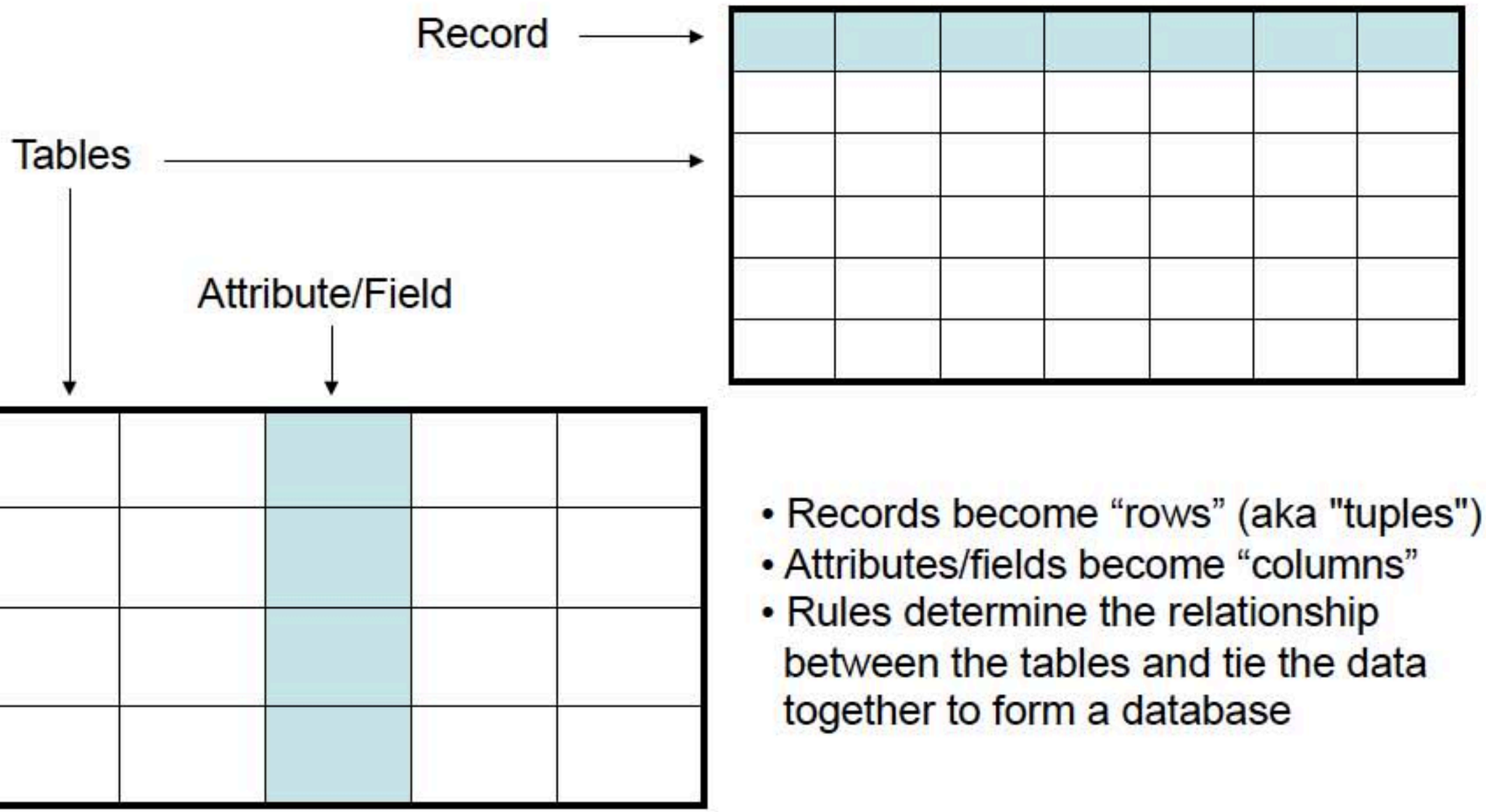
What is a relational database?

- Originally developed by E.F. Codd in 1970
- Organizes data into tables where each item is a row and the attributes of the item are in columns.
- Different from “flat file” databases because you can define “relationships” between items in different tables.
- The data within tables in the same database should all be related somehow.

Parts of a database

- Attributes (fields)
  - An attribute or field is a component of a record that describes something about an item.
- Records
  - A record is the representation of an individual item.
- Table
  - A collection of records
- Database
  - A collection of tables and rules for accessing the tables

Parts of a database



- Records become “rows” (aka "tuples")
- Attributes/fields become “columns”
- Rules determine the relationship between the tables and tie the data together to form a database

Database engines

- There are a variety of relational database management systems, or RDMSs
  - MS Access, ORACLE, Sybase, MySQL, Postgres are some of the more common ones
  - We use MySQL: free, robust, well documented, well maintained
- These generally operate as client-server systems and allow programmatic access
  - e.g. with Perl, Java, C, R, etc.
- SQLite is a light-weight RDMS
  - you can use it locally with R, Bioconductor (NOT a client-server system)
  - does not have all the "bells and whistles" of a full-blown RDMS.

What is SQL?

Structured Query Language

The (almost) universal database-speak, recognized (in some dialect) around the world.

Origin: SEQUEL (Structured English Query Language), developed in the System-R project of IBM (1974)

How is SQL used?

- The user specifies a certain condition.
- The program will go through all the records in the database file and select those records that satisfy the condition (searching).
- The result of the query will then be stored in form of a table.
- Results may be collations, specific combinations, subsets, and/or statistical information of the data.

SQL is a Standard - BUT....

- SQL is an ANSI (American National Standards Institute) standard computer language for accessing and manipulating database systems.

- SQL statements are used to retrieve and update data in a database.

- SQL works with database programs like MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase, etc.

- Unfortunately, there are many different versions of the SQL language, but to be in compliance with the ANSI standard, they must support the same major keywords in a similar manner (such as SELECT, UPDATE, DELETE, INSERT, WHERE, and others).

- Most of the SQL database programs also have their own proprietary extensions in addition to the SQL standard!

SQL Database Tables

A database most often contains multiple tables.

Each table is identified by a name (e.g. "Customers" or "Orders").

Tables contain records (rows) with data.

Below is an example of a table called "Person":

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

SQL Queries

With SQL, we can query a database and have a result set returned.
A query like this:

SELECT LastName FROM Person

Gives a result set like this:

LastName
Hansen
Svendson
Pettersen

Basic SQL Commands

- Creating tables with CREATE
- Adding data with INSERT
- Viewing data with SELECT
- Removing data with DELETE
- Modifying data with UPDATE
- Destroying tables with DROP

SQL Data Languages

- The Structured Query Language provides a syntax for executing queries

- The SQL language also includes syntax for:
  - DEFINING database tables and database constraints
  - MANIPULATING data records in database tables

SQL Data Definition Language (DDL)

The Data Definition Language (DDL) part of SQL permits database tables to be created or deleted.

We can also define indexes (keys), specify links between tables, and impose constraints between database tables.

The most important DDL statements in SQL are:

- **CREATE TABLE** - creates a new database table
- **ALTER TABLE** - alters (changes) a database table
- **DROP TABLE** - deletes a database table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

SQL Data Manipulation Language (DML)

These query and update commands together form the Data Manipulation Language (DML) part of SQL:

- **SELECT** - extracts data from a database table
- **UPDATE** - updates data in a database table
- **DELETE** - deletes data from a database table
- **INSERT INTO** - inserts new data into a database table

Creating tables with CREATE

- Generic form

```
CREATE TABLE tablename (
    column_name data_type attributes…,
    column_name data_type attributes…,
    …
)
```

- Table and column names can't have spaces or be "reserved words" like TABLE, CREATE, etc.

Data Types

- Binary
  - Database specific binary objects (BLOB)
- Boolean
  - True/False values (BOOLEAN)
- Character
  - Fixed width (CHAR) or variable size (VARCHAR)
- Numeric
  - Integer (INT), Real (FLOAT), Money (MONEY)
- Temporal
  - Time (TIME), Date (DATE), Timestamp (TIMESTAMP)

Phone Book/Contact Record

Name	Character
Address	Character
Company	Character
Phone Number	Character
URL/Web Page	Character
Age	Integer
Height	Real (float)
Birthday	Date
When we added the entry	Timestamp

Phone Book/Contact Table

```
CREATE TABLE contacts (
    Name                VARCHAR(40),
    Address             VARCHAR(60),
    Company             VARCHAR(60),
    Phone               VARCHAR(11),
    URL                 VARCHAR(80),
    Age                 INT,
    Height              FLOAT,
    Birthday            DATE,
    WhenEntered         TIMESTAMP
);
```

Plan your tables very carefully!
Once created, changing them can be tricky!

Phone Book/Contact Table

```
CREATE TABLE contacts (
    ContactID           INT AUTO_INCREMENT,
    Name                VARCHAR(40),
    Address             VARCHAR(60),
    Company             VARCHAR(60),
    Phone               VARCHAR(11),
    URL                 VARCHAR(80),
    Age                 INT,
    Height              FLOAT,
    Birthday            DATE,
    WhenEntered         TIMESTAMP,
    PRIMARY KEY (ContactID)
);
```

- ✓ *If you are going to use the relational nature of a database, don't forget you need to have a unique way to access records.*
- ✓ *There is a way to make the key automatically increment, so you don't have to worry about which one is next.*

The INSERT INTO Statement

The INSERT INTO statement is used to insert new rows into a table.

Syntax

```
INSERT INTO table_name
VALUES (value1, value2,....) ;
```

You can also specify the columns for which you want to insert data:

```
INSERT INTO table_name (column1, column2,...)
VALUES (value1, value2,....) ;
```

Insert a New Row

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger

And this SQL statement:

```
INSERT INTO Person
VALUES ('Hetland', 'Camilla', 'Hagabakka 24', 'Sandnes');
```

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes

Insert Data in Specified Columns

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes

And This SQL statement:

```
INSERT INTO Person (LastName, Address)
VALUES ('Rasmussen', 'Storgt 67');
```

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes
Rasmussen		Storgt 67	

The Update Statement

The UPDATE statement is used to modify the data in a table.

Syntax

```
UPDATE table_name
SET column_name = new_value
WHERE column_name = some_value ;
```

Update one Column in a Row

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen		Storgt 67	

We want to add a first name to the person with a last name of "Rasmussen":

```
UPDATE Person SET FirstName = 'Nina'
WHERE LastName = 'Rasmussen' ;
```

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Storgt 67	

Update several Columns in a Row

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen		Storgt 67	

We want to change the address and add the name of the city:

```
UPDATE Person
SET Address = 'Stien 12', City = 'Stavanger'
WHERE LastName = 'Rasmussen' ;
```

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Stien 12	Stavanger

The Delete Statement

The DELETE statement is used to delete rows in a table.

Syntax

```
DELETE FROM table_name
WHERE column_name = some_value ;
```

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Stien 12	Stavanger

"Nina Rasmussen" is going to be deleted:

DELETE FROM Person WHERE LastName = 'Rasmussen'

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger

Delete All Rows

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM table_name ;

Or

DELETE * FROM table_name ;

Or

TRUNCATE table_name ;

Destroying tables with DROP

Syntax

DROP TABLE *tablename*;

Example:

DROP TABLE contacts;

DROP TABLE Person;

Viewing data with SELECT

A SELECT query in SQL can consist of up to six clauses, but only the first two, SELECT and FROM, are mandatory.

The clauses are specified in the following order:

SELECT <attribute list>
FROM <table list>
[WHERE <condition>]
[GROUP BY <grouping attribute(s)>]
[HAVING <group condition>]
[ORDER BY <attribute list>]

- The most used command
- Probably the most complicated also
- If used improperly, can cause very long waits because complex computations

Basic structure of an SQL query

General Structure	SELECT, ALL / DISTINCT, *, AS, FROM, WHERE
Comparison	IN, BETWEEN, LIKE "% _"
Grouping	GROUP BY, HAVING, COUNT( ), SUM( ), AVG( ), MAX( ), MIN( )
Display Order	ORDER BY, ASC / DESC
Logical Operators	AND, OR, NOT
Output	INTO OUTFILE [ADDITIVE], LIMIT [offset,] rows
Union	UNION

Summary of SQL SELECT

The SELECT clause

lists the attributes or functions to be retrieved

The FROM clause

specifies all relations (or aliases) needed in the query but not those needed in nested queries

The WHERE clause

specifies the conditions for selection and join of tuples from the relations specified in the FROM-clause

GROUP BY

specifies grouping attributes

HAVING

specifies a condition for selection of groups

ORDER BY

specifies an order for displaying the result of a query

A query is evaluated by first applying the WHERE-clause, then GROUP BY and HAVING, and finally the SELECT-clause

A few simple SELECTs

SELECT * FROM contacts;

- *Display all records in the 'contacts' table*

SELECT contactid,name FROM contacts;

- *Display only the record number and names*

SELECT DISTINCT url FROM contacts;

- *Display only one entry for every value of URL*

Refining selections with WHERE

The WHERE "subclause" allows you to select records based on a condition.

```
SELECT * FROM contacts
        WHERE age<10;
```

– *Display records from contacts where age<10*

```
SELECT * FROM contacts
        WHERE age BETWEEN 18 AND 35;
```

– *Display records where age is 18-35*

WHERE operators

With the WHERE clause, the following operators can be used:

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern

Note: In some versions of SQL the <> operator may be written as !=

Another WHERE Example

To select only the persons living in the city "Sandnes", we add a WHERE clause to the SELECT statement:

```
SELECT * FROM Person
WHERE City='Sandnes';
```

LastName	FirstName	Address	City	Year
Hansen	Ola	Timoteivn 10	Sandnes	1951
Svendson	Tove	Borgvn 23	Sandnes	1978
Svendson	Stale	Kaivn 18	Sandnes	1980
Pettersen	Kari	Storgt 20	Stavanger	1960

LastName	FirstName	Address	City	Year
Hansen	Ola	Timoteivn 10	Sandnes	1951
Svendson	Tove	Borgvn 23	Sandnes	1978
Svendson	Stale	Kaivn 18	Sandnes	1980

The LIKE Condition

The LIKE condition is used to specify a search for a pattern in a column.

Syntax

SELECT column FROM table
WHERE column LIKE pattern ;

A "%" sign can be used to define wildcards (missing letters in the pattern) both before and after the pattern.

SELECT * FROM contacts
WHERE name LIKE 'J%';
- Display records where the name starts with 'J'

SELECT * FROM contacts
WHERE url LIKE '%.com';
- Display records where url ends in ".com"

Using Quotes

Note that we have used single quotes around the conditional values in the examples.

SQL uses single quotes around text values (most database systems will also accept double quotes). Numeric values should not be enclosed in quotes.

This is correct:

```
SELECT * FROM Persons WHERE FirstName='Tove' ;
```

This is wrong:

```
SELECT * FROM Persons WHERE FirstName=Tove ;
```

Null Values

- Field values in a tuple are sometimes *unknown* (e.g., a rating has not been assigned) or *inapplicable* (e.g., no spouse's name).
  - SQL provides a special value *null* for such situations.

- The presence of *null* complicates many issues. E.g.:
  - Special operators needed to check if value is/is not *null*.
  - Is *rating>8* true or false when *rating* is equal to *null*? What about AND, OR and NOT connectives?
  - We need a 3-valued logic (true, false and *unknown*).
  - Meaning of constructs must be defined carefully. (e.g., WHERE clause eliminates rows that don't evaluate to true.)
  - New operators (in particular, *outer joins*) possible/needed.

More SELECT examples

```
SELECT * FROM contacts
        WHERE name IS NULL;
```

- *find something with a null value*

```
SELECT * FROM contacts
        WHERE zip IN ('14454','12345');
```

- *find something that matches one of multiple values*

```
SELECT * FROM contacts
        WHERE zip IN (
            SELECT zip FROM address
                    WHERE state='NY'
            );
```

- *a SUBSELECT: find something that matches a result from another SELECT statement*

ORDER BY

- The “ORDER BY” clause allows you to sort the results returned by SELECT.
- You can order in ASCending or DESCending order (ASC is default).
- You can order on one or more columns.

```
SELECT * FROM contacts
   ORDER BY company DESC;

SELECT * FROM contacts
   ORDER BY company, name;

SELECT * FROM Person
   ORDER BY LastName, FirstName;
```

GROUP BY/HAVING

- The “GROUP BY” clause allows you to group results together with “aggregate functions”
  - AVG(), COUNT(), MAX(), MIN(), SUM()
  - COUNT DISTINCT

- HAVING allows you to search the GROUP BY results

GROUP BY Examples

```
SELECT company,count(company)
   FROM contacts
   GROUP BY company;
```

– *list each company along with # of contacts at that company*

```
SELECT company,count(company)
   FROM contacts
   GROUP BY company
   HAVING count(company) > 5;
```

– *list companies with at least 5 contacts and the # of contacts*

Aggregation

GROUP BY is an aggregator clause. SQL supports several other aggregation operations:

SUM, MIN, MAX, AVG, COUNT

Except COUNT, all aggregations apply to a single attribute:

```
SELECT  COUNT(*)
FROM    Purchase ;
```

```
SELECT  MIN(price), MAX(price), AVG(price)
FROM    Product
WHERE   manufacturer='Toyota' ;
```

Using Aliases

Sometimes it's convenient to use an ALIAS to refer to a table, a column, or the result of an calculation. You can do this using 'AS':

```
SELECT  min(price) AS Cheapest,
        max(price) AS Most_Expensive,
        avg(price) AS Average
FROM    Product
WHERE   manufacturer='Toyota' ;
```

Aliases are used so often that the first word after a table name, column, or calculation is recognized as an alias in a SELECT statement, even if you forget to include 'AS' explicitly.

We'll see more examples later.

The SELECT DISTINCT Statement

The SELECT statement returns information from table columns. But what if we only want to select distinct elements?

With SQL, all we need to do is to add a DISTINCT keyword to the SELECT statement in order to return a set of distinct (unique / different) values:

Syntax

```
SELECT DISTINCT column_name(s)
FROM table_name ;
```

Using the DISTINCT keyword

Orders	
Company	**OrderNumber**
Sega	3412
W3Schools	2312
Trio	4678
W3Schools	6798

To select ALL values from the column "Company" we use a SELECT statement like this:

```
SELECT Company FROM Orders ;
```

→

Company
Sega
W3Schools
Trio
W3Schools

Note that "W3Schools" is listed twice in the result-set.

To select only DIFFERENT values from the column named "Company" we use a SELECT DISTINCT statement like this:

```
SELECT DISTINCT Company
FROM Orders ;
```

→

Company
Sega
W3Schools
Trio