

# DB Project

---

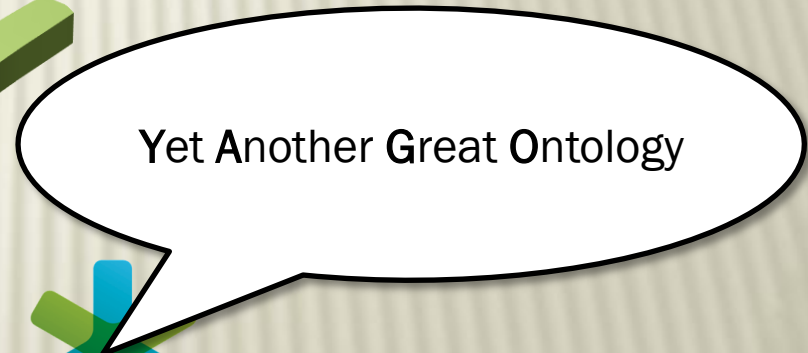
Database Systems  
Spring 2013

# Database project – YAGO



**WIKIPEDIA**  
*The Free Encyclopedia*

**WordNet**  
A lexical database for English



**yago**  
select knowledge

# About YAGO\*

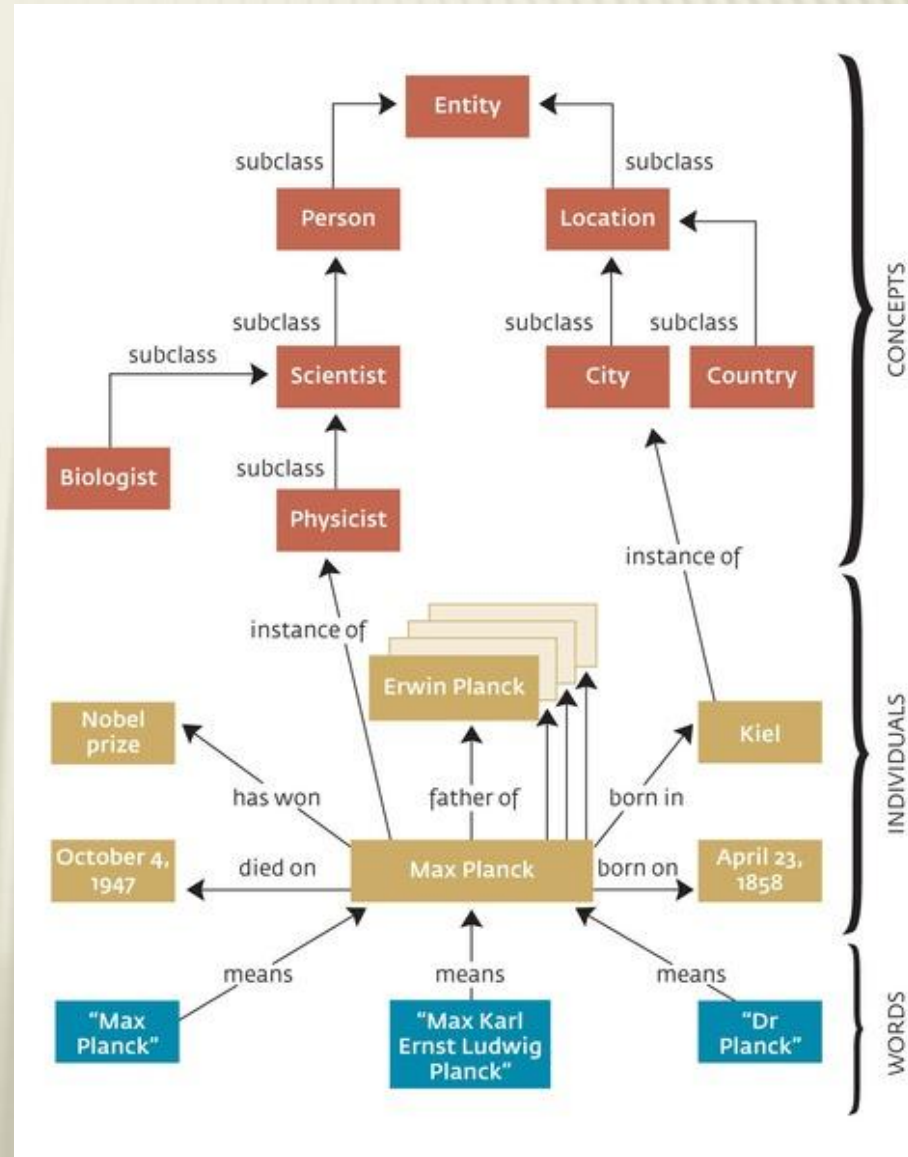
---

- × A huge **semantic** knowledge base
    - + **knowledge base** – a special kind of DB for knowledge management (e.g., facts)
    - + **Semantic** – related to the Semantic Web where presentation is assigned a meaning
- [http://www.youtube.com/watch?v=TJfrNo3Z-DU&feature=player\\_embedded](http://www.youtube.com/watch?v=TJfrNo3Z-DU&feature=player_embedded)



# YAGO is an **Ontology**

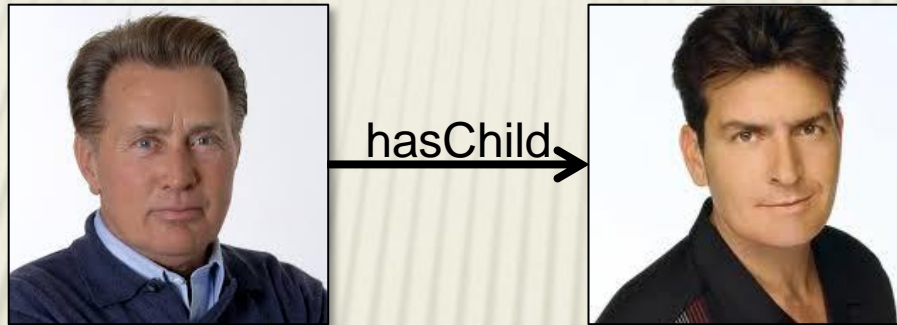
- ✖ A Taxonomy of concept classes
- ✖ At the bottom – instances, facts about instances
  - + This is typically the interesting part



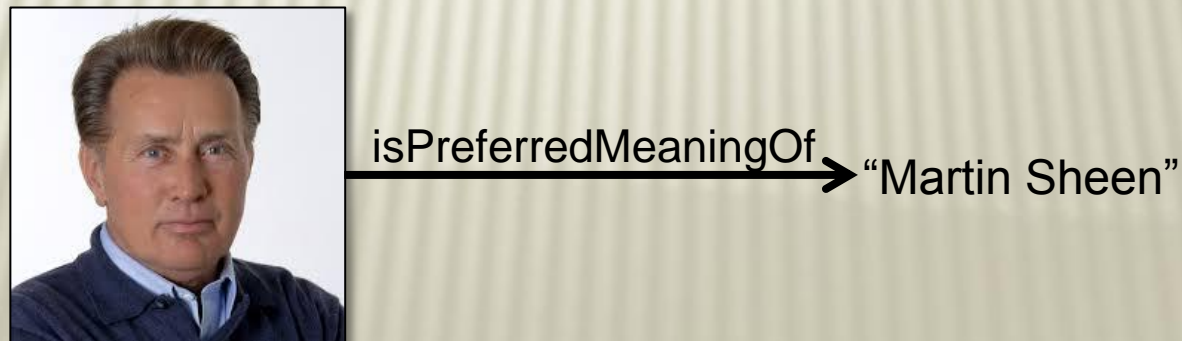


# A fact

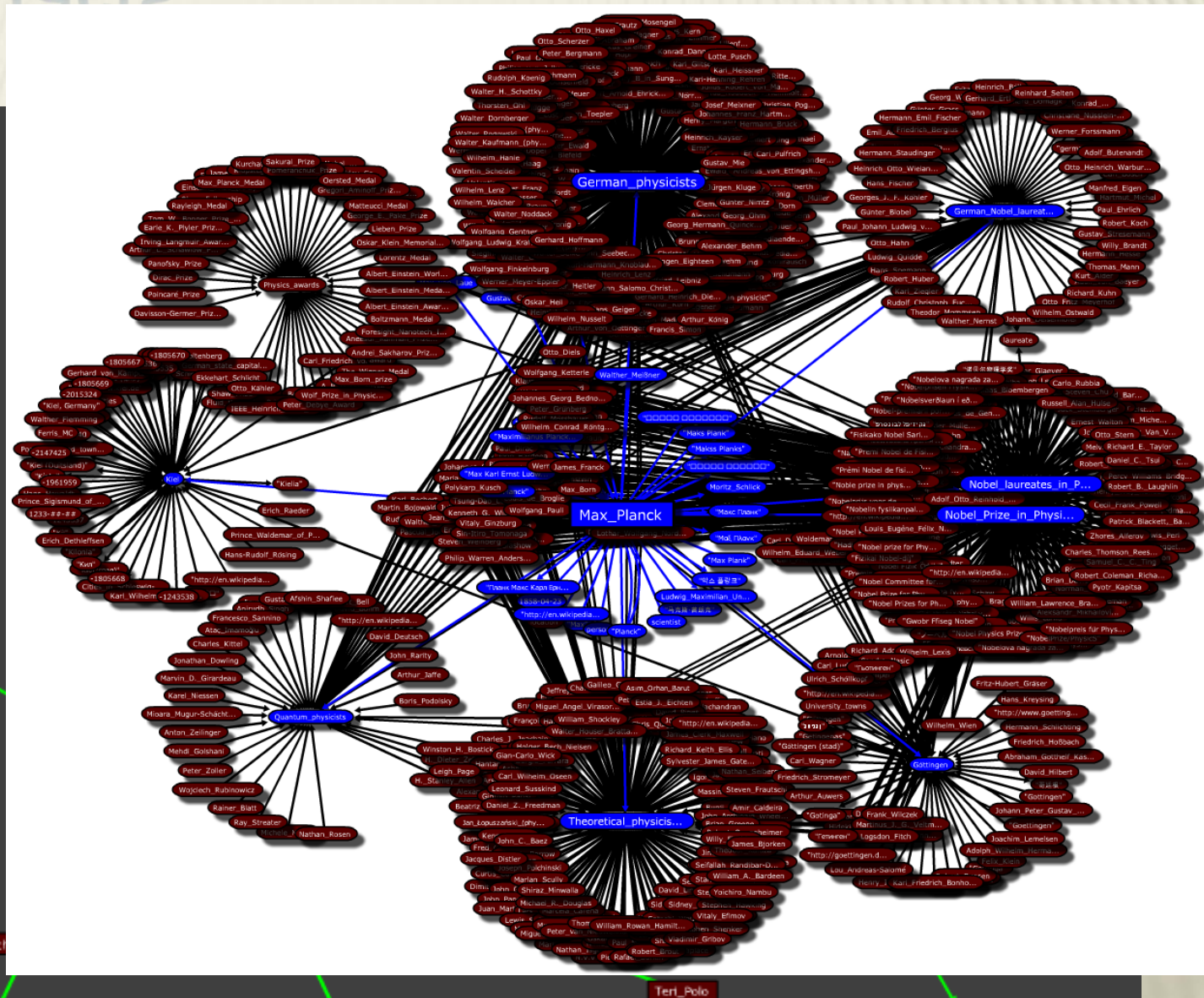
- ✗ A particular relationship that holds between two instances



- ✗ Or, an instance and a literal (string)



# Graph of facts



# More about YAGO

- × More than 10 million entities
- × More than 120 million facts
- × High (but not perfect!) accuracy
- × Connections with other ontologies (DBPedia, SUMO, Freebase...)
- × Over 11 research papers (Max Planck team) with over 1k citations

## Detailed Evaluation Results of YAGO2s

Relation	Evaluations	Correct	Ratio (%)	Wilson Center (%)	Wilson Width (%)	# of facts in YAGO
<actedIn>	39	39	100	95.52	4.48	120508
<created>	55	54	98.18	95.04	4.64	260997
<dealsWith>	43	43	100	95.9	4.1	882
<diedIn>	35	35	100	95.05	4.95	45019
<diedOnDate>	37	37	100	95.3	4.7	361890
<directed>	58	57	98.28	95.28	4.42	39685
<edited>	35	35	100	95.05	4.95	5678
<exports>	88	88	100	97.91	2.09	577
<graduatedFrom>	39	39	100	95.52	4.48	26280
<happenedIn>	36	36	100	95.18	4.82	166496
<happenedOnDate>	43	43	100	95.9	4.1	182572
<hasAcademicAdvisor>	44	44	100	95.99	4.01	2895
<hasArea>	42	42	100	95.81	4.19	129909
<hasBudget>	44	44	100	95.99	4.01	715
<hasCapital>	63	61	96.83	94.13	4.99	1821
<hasChild>	64	62	96.88	94.22	4.92	14998
<hasCurrency>	36	36	100	95.18	4.82	504

# Database Project - Goals

- × Project goal: to tackle and resolve **real-life** DB related development issues
- × Including
  - × DB design
  - × Query writing
  - × DB programming
  - × Application design





# Database Project - Requirements

1. Think of an application
  - + Useful and creative!
2. Design a DB schema
  - + According to available data
  - + And the application usage
  - + And principles of DB design
3. Load *and flatten* data from YAGO
4. Update the Database
5. Write an application (**with UI**)
  - + Usable and fault tolerant
  - + Accessing the data via efficient queries/updates
  - + According to principles of coding
6. Support **manual updates** and **updates from YAGO**

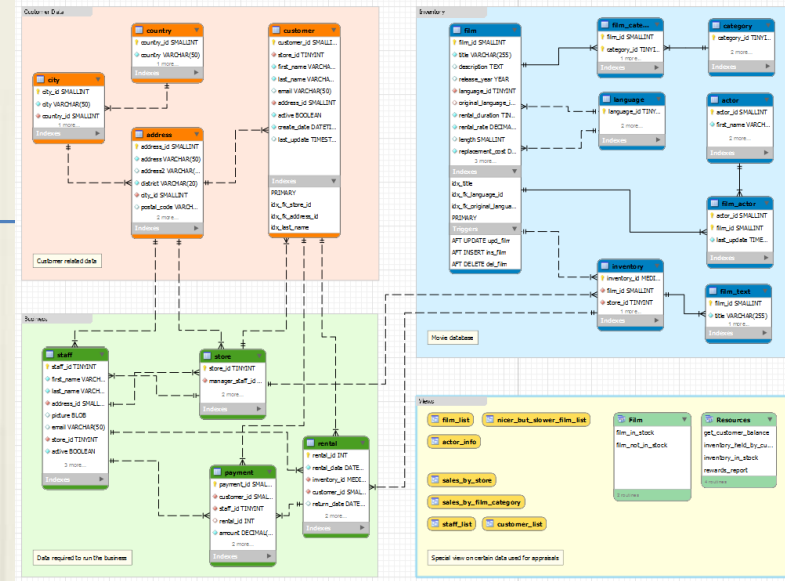
# 1. Think of an application

- ✗ **Could be anything!** As far as your imagination goes
  - + YOU should want to use it...
  - + **Tip:** first inspect the available data
  - + **Tip:** must-have and nice-to-have features
  - + The application can be interesting even if the UI is simple



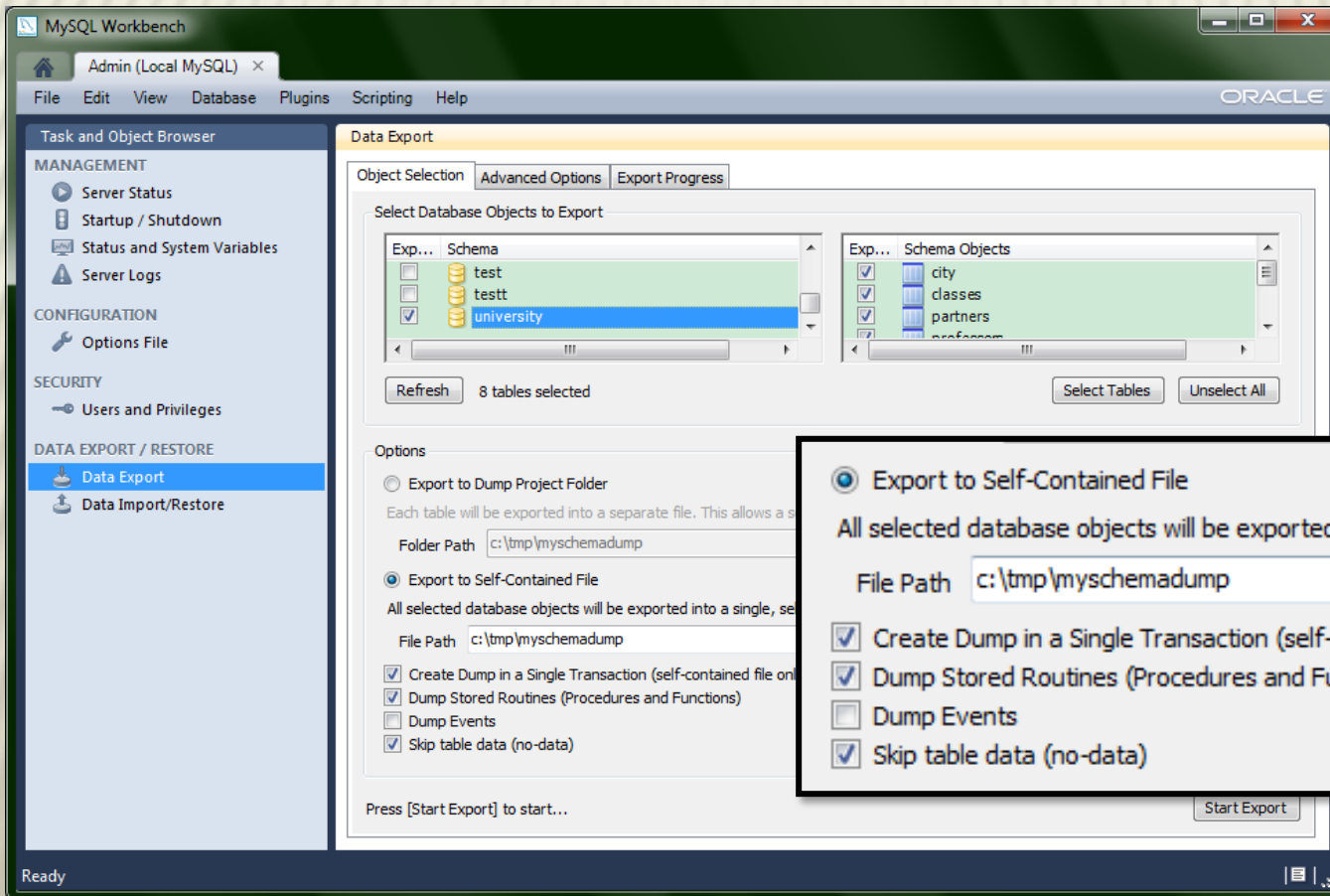
## 2. Design a DB Schema

- ✖ Tables, indexes, keys and foreign keys
  - ✖ Avoid redundant information
  - ✖ Allow efficient queries
  - ✖ The script for generating the schema should be submitted with the project
- 
- ✖ More about design in the following lectures



## 2. Creating the SQL Script

- ✗ <http://dev.mysql.com/doc/workbench/en/wb-manage-server-export-to-disk.html>







### 3. Load data from YAGO

- ✗ The entire database of YAGO is freely available online
- ✗ Extract relevant parts (entities, facts)
- ✗ Insert into *flat tables*
  - + A few facts may be used for one record
  - + E.g., the actor record for Martin Sheen will include his first and last name, birth date, residence, etc.
  - + (But not the films he did... why?)
- ✗ We discuss this in detail next

## 4. Update the DB

- ✗ The data should be written to the DB
  - + Before submission you will update your schema in the school MySQL server
- ✗ Including relevant IDs
  - + Actor\_id, film\_id,... (**Must be integers in MySQL!**)
  - + Auto-incremental or based on YAGO ids

Column Name	Datatype	P..	N.	U..	B..	U.	Z..	AI	Default
student_id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Auto Incremental
student_name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
city_id	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

## 5. Write an application

---

- ✗ In java, using JDBC
- ✗ Desktop application
- ✗ SWT for GUI (other open-source packages such as Swing, Qt Jambi...)
- ✗ Any other open-source packages,  
*except hibernate and similar packages*
- ✗ According to DB programming principles
- ✗ **Important:** separate the code of the UI, the core logic and the DB

## 5. Write an application (cont.)

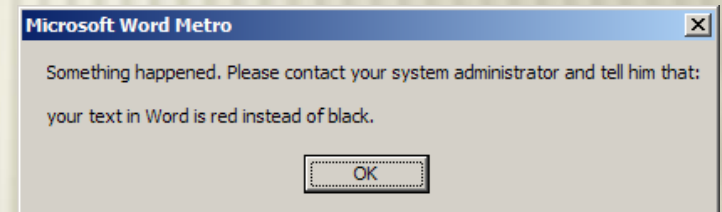
---

- ✖ Using the DB data
- ✖ Efficient queries / updates
  - + Important for user experience
  - + Use indexes!
- ✖ Interesting queries / updates
  - + Search for specific data
  - + According to your application



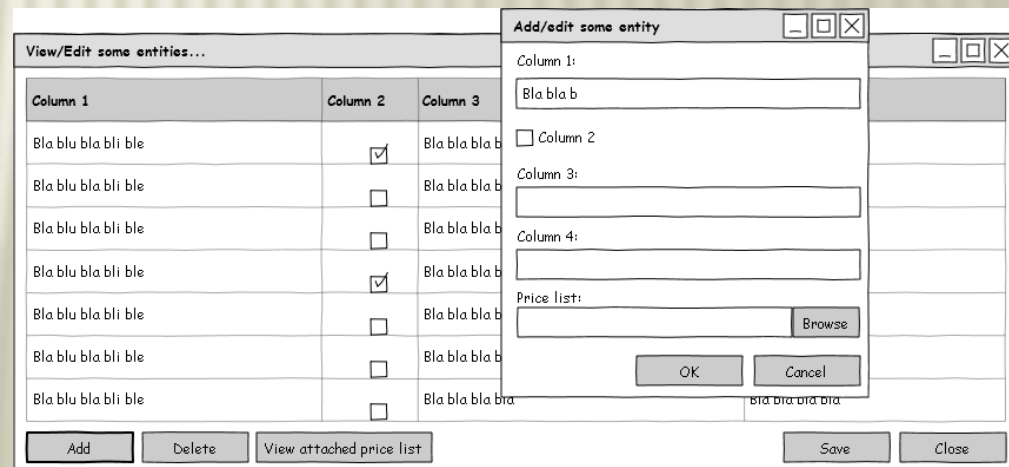
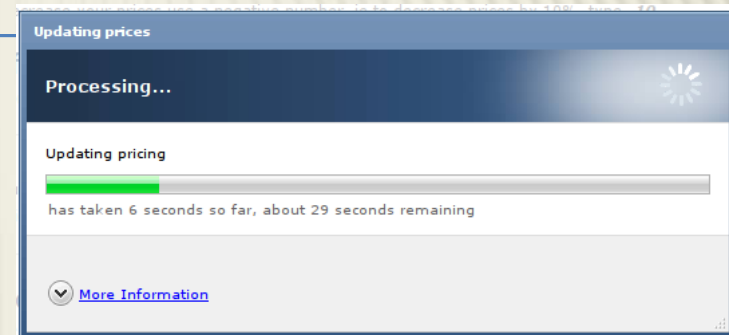
## 5. Write an application (cont.)

- ✗ Should be usable and easy to understand
- ✗ Should be fault tolerant
  - + Every exception should be caught, and a user-friendly message should be displayed
- ✗ Test your application
  - + Install on different environments
  - + Portable:
    - ✗ Copy-paste, create DB schema, edit configuration and... play!



## 6. Support updates

- ✗ A must-have feature!
- ✗ “Import” from YAGO
  - + Via the UI
  - + To support, e.g., a new YAGO version
  - + What happens to the “old” data?
  - + Administrator privileges?
- ✗ Manual updates
  - + Add, edit and delete data originally taken from YAGO
  - + Add, edit and delete user-provided data



# In the course Website

---

- × Project details
- × Project examination form and grade guide

<http://courses.cs.tau.ac.il/databases/databases201213b/assignments/>

# What to focus on

---

- × Database structure
- × Data – you choose what to take from YAGO
- × Query efficiency
- × Editing capabilities
- × Usability and fault tolerance



# YAGO data – HowTo

✖ YAGO downloads page - <http://www.mpi-inf.mpg.de/yago-naga/yago/downloads.html>

TAXONOMY	<b>yagoSchema</b> The domains, ranges and confidence values of the relations	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
	<b>yagoTypes</b> All rdf:type facts of YAGO	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
	<b>yagoTaxonomy</b> The entire YAGO taxonomy. These are all rdfs:subClassOf facts derived from Wikipedia and from WordNet.	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
	<b>yagoTransitiveType</b> Transitive closure of all rdf:type/rdfs:subClassOf facts	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
SIMPLETAX	<b>yagoSimpleTaxonomy</b> A simplified rdfs:subClassOf taxonomy. This taxonomy contains just WordNet leaves, the main YAGO branches, and owl:Thing. Use with yagoSimpleTypes.	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
	<b>yagoSimpleTypes</b> A simplified rdf:type system. This theme contains all instances, and links them with rdf:type facts to the leaf level of WordNet. Use with yagoSimpleTaxonomy.	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
CORE	<b>yagoFacts</b> All facts of YAGO that hold between instances	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
	<b>yagoLabels</b> All facts of YAGO that contain labels (rdfs:label, skos:prefLabel, isPreferredMeaningOf, hasGivenName, hasFamilyName, hasGloss)	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
	<b>yagoLiteralFacts</b> All facts of YAGO that contain literals (except labels)	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
GEONAMES	<b>yagoGeonamesClassIds</b> IDs from GeoNames classes	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
	<b>yagoGeonamesClasses</b> Classes from GeoNames	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
	<b>yagoGeonamesEntityIds</b> IDs from GeoNames entities	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>
META	<b>yagoStatistics</b> Statistics about YAGO and YAGO themes	<a href="#">Preview</a>	<a href="#">Download TTL</a>	<a href="#">Download TSV</a>

# YAGO data – HowTo (cont.)

- ✗ Data comes in TSV format – text with tab-separated fields (also TTL)

**Format:**    yago-id                      entity                      relation                      entity

<id_zik11d_88c_ehg9uq>	<A>	rdf:type	<wikicategory_Vowel_letters>
<id_zik11d_88c_w3c6wm>	<A>	rdf:type	<wikicategory_ISO_basic_Latin_letters>
<id_1bsrlah_88c_1s6g79w>	<Alabama>	rdf:type	<wikicategory_States_of_the_United_States>
<id_3ienox_88c_4retae>	<Achilles>	rdf:type	<wikicategory_People_of_the_Trojan_War>
<id_3ienox_88c_1rk49a2>	<Achilles>	rdf:type	<wikicategory_Pederastic_heroes_and_deities>
<id_3ienox_88c_s57m6o>	<Achilles>	rdf:type	<wikicategory_Kings_of_the_Myrmidons>

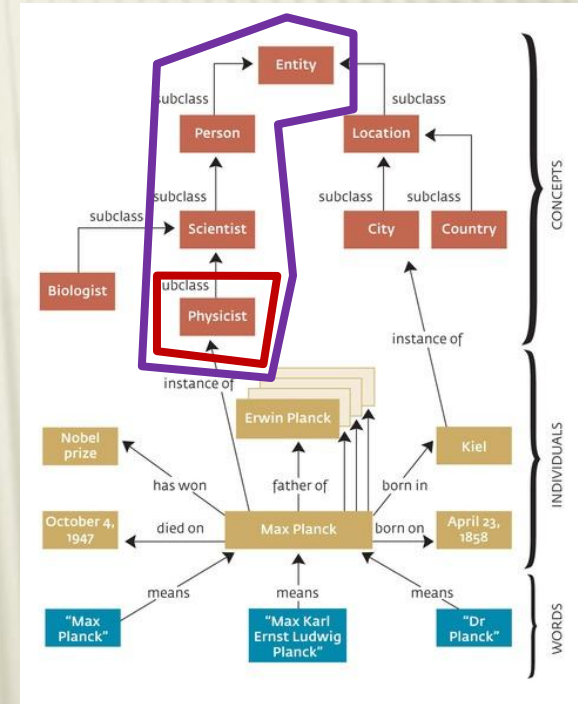
- ✗ YAGO entities and relations are marked by < > (e.g., <Achilles>)
- ✗ Others are taken from rdf, rdfs, owl, skos... (e.g., rdf:type)
- ✗ Literals are marked by " "
  - + Strings with optional locale, e.g., "Big tent"@eng
  - + Others with datatype, e.g., "1977-08-16"^^xsd:date, "70"^^<m>
- ✗ See also: <http://www.mpi-inf.mpg.de/yago-naga/yago/faq.html>

# YAGO data – HowTo (cont.)

- ✖ You can also download just the portions of YAGO2s that you need. Each portion is called a theme. There are 8 groups of themes:
  - + TAXONOMY: All types of entities, and the class structure of YAGO2s. Moreover, it has formal definitions of YAGO relations.
  - + SIMPLETAX: An alternative, simpler taxonomy of YAGO.
  - + CORE: Core facts of YAGO2s, such as the facts between entities, the facts containing literals, i.e., numbers, dates, strings, etc.
  - + GEONAMES: Geographical entities, classes taken from GeoNames.
  - + META: Temporally and spatially scoped facts together with statistics and extraction sources about the facts.
  - + MULTILINGUAL: The multilingual names for entities.
  - + LINK: The connection of YAGO2s to Wordnet, DBPedia, etc.
  - + OTHER: Miscellaneous features of YAGO2s, such as Wikipedia in-outlinks, GeoNames data etc.

# YAGO data – Taxonomy

- ✗ **yagoTypes** – facts with relation `rdf:type` - contains the lowest-level classes for each entity
- ✗ **yagoTransitiveType** – also contains the higher-level classes



<id_zik11d_88c_ehg9uq>	<A>	<b>rdf:type</b>	<wikicategory_Vowel_letters>
<id_zik11d_88c_w3c6wm>	<A>	<b>rdf:type</b>	<wikicategory_ISO_basic_Latin_letters>
<id_1bsrlah_88c_1s6g79w>	<Alabama>	<b>rdf:type</b>	<wikicategory_States_of_the_United_States>
<id_3ienox_88c_4retae>	<Achilles>	<b>rdf:type</b>	<wikicategory_People_of_the_Trojan_War>
<id_3ienox_88c_1rk49a2>	<Achilles>	<b>rdf:type</b>	<wikicategory_Pederastic_heroes_and_deities>
<id_3ienox_88c_s57m6o>	<Achilles>	<b>rdf:type</b>	<wikicategory_Kings_of_the_Myrmidons>



# YAGO data - Core

---

- ✖ yagoFacts – facts between instances

- + A complete list of relations – in Taxomony, yagoSchema  
<Martin\_Sheen> <hasChild> <Charlie\_Sheen>

- ✖ YagoLabels – names of entities.

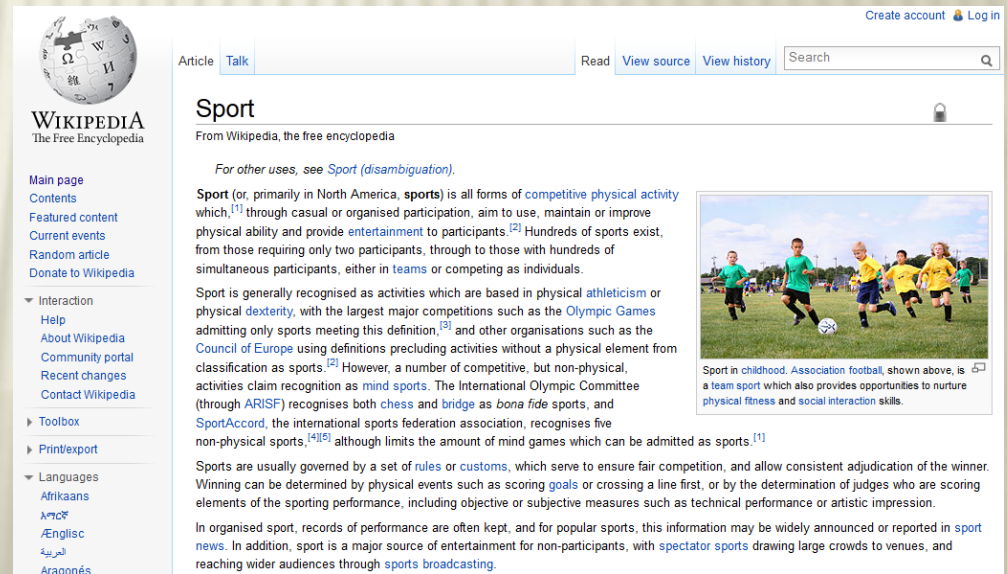
- + There may be many labels! use skos:prefLabel  
<Martin\_Sheen> skos:prefLabel "Martin Sheen"@eng

- ✖ yagoLiteralFacts – other facts with literals

- + Often properties of the entity  
<Martin\_Sheen> <wasBornOnDate> "1940-08-03"^^xsd:date

# Example

- ✖ Assume we work with the sports domain
- ✖ Create an online application that contains details on teams and players
- ✖ Users/automatic algorithms will guess game scores, awards, etc.



The screenshot shows the Wikipedia article for "Sport". The page layout includes a sidebar on the left with navigation links such as "Main page", "Contents", "Featured content", "Current events", "Random article", "Donate to Wikipedia", "Interaction", "Help", "About Wikipedia", "Community portal", "Recent changes", "Contact Wikipedia", "Toolbox", "Print/export", "Languages", and "Afrikaans". The main content area has a title "Sport" and a subtitle "From Wikipedia, the free encyclopedia". Below the title, there is a note: "For other uses, see [Sport \(disambiguation\)](#)." The article text begins with a definition: "Sport (or, primarily in North America, **sports**) is all forms of competitive physical activity which, through casual or organised participation, aim to use, maintain or improve physical ability and provide entertainment to participants." It then discusses the recognition of sports by various organizations, including the Olympic Games, the Council of Europe, and the International Olympic Committee. A photograph of children playing soccer is included, with a caption: "Sport in childhood. Association football, shown above, is a team sport which also provides opportunities to nurture physical fitness and social interaction skills." The article continues with a discussion of the rules and customs governing sports, and the role of sports in organized competition and entertainment.

# Example



The image shows a digital scoreboard with two main sections: 'SUNDAY, OCT 11, 2009 N.F.L. WEEK #5' and 'COLLEGE FOOTBALL SATURDAY, OCT 10, 2009'. The NFL section lists teams 401 through 412 with their respective odds and bet amounts. The College Football section lists games 319 through 324 with team names and odds.

SUNDAY, OCT 11, 2009 N.F.L. WEEK #5			COLLEGE FOOTBALL SATURDAY, OCT 10, 2009		
401 VIKINGS	-10	-600313	313 E. MICH.		
402 RAMS	41	+400314	314 CENT MICH	-23	
403 COWBOYS	-8	-400315	315 DUKE		
404 CHIEFS	42.5	+300316	316 N C STATE	-16.5	
405 REDSKINS	37.5	+165317	317 MARYLAND		
406 PANTHERS	-3.5	-185318	318 WK FOREST	-12.5	
407 BUCCANEER			319 INDIANA		
408 EAGLES	-14.5		320 VIRGINIA	-7	
409 RAIDERS			321 PURDUE		
410 GIANTS	-15		322 MINN.	-3	
411 BROWNS	41	+220323	323 U CONN	46	
412 BILLS	-6	-300324	324 PITT.	-6.5	

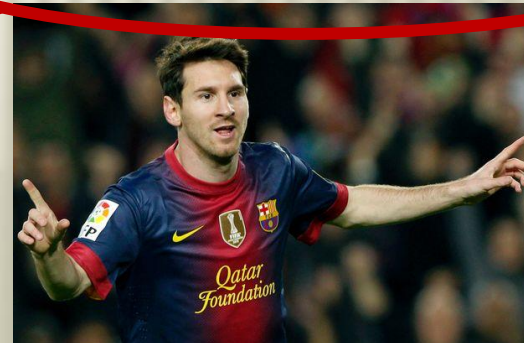
- ✖ Editing capabilities for YAGO data:  
add/remove/edit all players, teams, games...
- ✖ Data of your own: odds, bets...
- ✖ Your tables:
  - + Players, Teams, Users, Bets
  - + Linking tables: Player\_team, User\_bets

# YAGO data – putting it together

- ✗ We want to create records in the table  
Player(ID, name, birth date, height)
- ✗ First, we look in yagoTransitiveType for entities  
that represent players
  - + We find, e.g.,

<Lionel_Messi>	rdf:type	<wordnet_player_110439851>
----------------	----------	----------------------------

Fixed  
application  
parameter





# YAGO data – putting it together



× Next, we create the properties

+ ID – e.g., automatically generated (must make sure we do not have Messi in our DB yet!)

+ Name – from yagoLabels

<Lionel_Messi>	skos:prefLabel	"Lionel Messi"@eng
----------------	----------------	--------------------

+ Birthdate and height – from yagoLiteralFacts

<Lionel_Messi>	<hasHeight>	"1.69"^^<m>
<Lionel_Messi>	<wasBornOnDate>	"1987-06-24"^^xsd:date



# YAGO data – Flattening process

---

1. Read the relevant TSV files
2. Save only the relevant data in memory or in a temporary table
3. Join together relevant pieces of data
4. Insert into the (final) schema tables

# YAGO data – challenges



- ✗ What do we do when a value is missing?
- ✗ What do we do when the data is invalid?
- ✗ What do we do when there is more than one value?

<Lionel_Messi>	<playsFor>	<FC_Barcelona_B>
<Lionel_Messi>	<playsFor>	<FC_Barcelona_C>
<Lionel_Messi>	<playsFor>	<Newell's_Old_Boys>
<Lionel_Messi>	<playsFor>	<Argentina_national_football_team>
<Lionel_Messi>	<playsFor>	<FC_Barcelona>

# Relaxations



- ✗ You do not have to fix errors in YAGO's data (but you can allow the application users to do so)
- ✗ You can choose an arbitrary value if there are many (where this makes sense! playsFor can be many-to-one, actedIn cannot)
- ✗ You can use an additional data source to complete missing data (must be freely available)

# Past years projects

SSDA Music Store Manager

Store Details

Store: 42 Sunday, 17/01/2010, 07:10  
Address: 321 Moshe Dayan St., Gyvataim  
Phone: 03-555-4321 Manager: 76543211

Welcome

Welcome to the SSDA Music Store Manager!  
Select your choice:  
Search for albums, place an order or add to sale, manage sales, manage stock (orders and requests) and manage HR and database

Search | Sale | Stock | Management

Search by

☐ Search by album ID:

☒ Search by other parameters:

☐ Album name:

☒ Artist:

☐ Year from:  To:

☒ Song name(s):

☒ Genre(s):  

☒ Rock ☐ Pop ☐ World ☐ Electronic

☐ Jazz ☐ Blues ☐ Hip-Hop ☐ Classical

☐ Other:

Stock: ☒ All ☐ In network ☐ In store

Clear Fields

Search

Stock Information

Store stock: 0  
Storage location: 0  
Price: 36  

Get Stock Information

Place an Order

Sale

Add to sale with quantity:  
  

Add to Sale...

Search Results

Album ID	Album Name	Artist	Year	Genre	Length
882849	electric magic o...	led zepp...	0	rock & roll	78:44
911119	dazed and con...	led zepp...	0	rock & roll	56:04
1264973	whole lotta fo...	led zepp...	1969	classic ro...	75:09
1151298	the sex machi...	led zepp...	1975	rock & roll	67:11

Show Song List

Track	Song name	Artist	Length
1	sweet baby	Various Artists	2:59
2	I can't quit you...	Various Artists	6:50
3	dazed and conf...	Various Artists	14:59
4	you shock me	Various Artists	10:42
5	how many mor...	Various Artists	10:45
6	hrra hrra hrra	Various Artists	11:42

33



# Past years projects







# Past years projects



# Past years projects

Music Marketplace

Search Album Info Requests and Offers Import

My Requests

Album Title	New
Luftslottet som sprengtes	
Rafi Resurrected	
Qurbani (remastered)	

Delete

Confirm Cancel all changes

My Offers

Album Title	New
Saturday Night Fever	
We Are... Sniffing Glue ....	

Delete

Username	Email	Can give you	Wants from you	Match
guy	guy.ba...	Qurbani (remaste...	Saturday Night Fever	!
guy	guy.ba...	Luftslottet som s...	Saturday Night Fever	!
guy	guy.ba...	Rafi Resurrected	Saturday Night Fever	!
guy	guy.ba...	Rafi Resurrected	Luftslottet som spre...	
guy	guy.ba...	Qurbani (remaste...	Saturday Night Fever	

# Tips

---

- \* First:
  - understand the data format.
  - understand what you want to do.
  - find relevant data and relations.
- \* Be flexible: work with what you have!
- \* Database key should always be an INTEGER.
- \* Don't forget to support manual edit of the data (add/update/remove) – e.g., artists/categories/values...
- \* Configuration – for DB connection, OS, etc.



# Database Project - Bureaucracy

---

- ✖ Hard work, but a practical experience.
- ✖ Work in groups of 4
- ✖ Submission database is MySQL in TAU
- ✖ Java, SWT (or Swing/AWT)
- ✖ Thinking out of the box will be rewarded




# Database Project - Requirements

---

- ✖ (at least) 150K records table
  - + But could be much more!
- ✖ Also see the course website for full instructions  
<http://courses.cs.tau.ac.il/databases/databases201213b/assignments/>

# Time schedule

---

- 
- **April 9<sup>th</sup>** – Project distribution
  - **April 18<sup>th</sup>** – Last date for submitting the team member names
  - **May 21<sup>st</sup>** – “Project days”
    - + I will meet with each group
    - + You need to prepare: DB design, preferably have data in the school DB, work plan – what is left to do, who does what and when, optional – presentation or demonstration
  - **June 18<sup>th</sup>** – Project due!
    - + Aim to submit a week before, to avoid network crashes, mysterious illnesses...

# DB Project

---

**בהצלחה!**