DB Project

Database Systems
Spring 2013
Database project – **YAGO**

**WordNet**
A lexical database for English

**Wikipedia**
The Free Encyclopedia

**Yet Another Great Ontology**

**GeoNames**

*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

* Fabian M. Suchanek, Gjergji Kasneci and Gerhard Weikum, YAGO - A Core of Semantic Knowledge, WWW ‘07
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)
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

<table>
<thead>
<tr>
<th>Relation</th>
<th>Evaluations</th>
<th>Correct</th>
<th>Ratio (%)</th>
<th>Wilson Center (%)</th>
<th>Wilson Width (%)</th>
<th># of facts in YAGO</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;actedIn&gt;</td>
<td>39</td>
<td>39</td>
<td>100</td>
<td>95.52</td>
<td>4.48</td>
<td>120508</td>
</tr>
<tr>
<td>&lt;created&gt;</td>
<td>55</td>
<td>54</td>
<td>98.18</td>
<td>95.04</td>
<td>4.64</td>
<td>260997</td>
</tr>
<tr>
<td>&lt;dealsWith&gt;</td>
<td>43</td>
<td>43</td>
<td>100</td>
<td>96.9</td>
<td>4.1</td>
<td>882</td>
</tr>
<tr>
<td>&lt;diedIn&gt;</td>
<td>35</td>
<td>35</td>
<td>100</td>
<td>95.05</td>
<td>4.95</td>
<td>45019</td>
</tr>
<tr>
<td>&lt;diedOnDate&gt;</td>
<td>37</td>
<td>37</td>
<td>100</td>
<td>95.3</td>
<td>4.7</td>
<td>361890</td>
</tr>
<tr>
<td>&lt;directed&gt;</td>
<td>58</td>
<td>57</td>
<td>98.28</td>
<td>95.28</td>
<td>4.42</td>
<td>39685</td>
</tr>
<tr>
<td>&lt;edited&gt;</td>
<td>35</td>
<td>35</td>
<td>100</td>
<td>95.05</td>
<td>4.95</td>
<td>5678</td>
</tr>
<tr>
<td>&lt;exports&gt;</td>
<td>88</td>
<td>88</td>
<td>100</td>
<td>97.91</td>
<td>2.09</td>
<td>577</td>
</tr>
<tr>
<td>&lt;graduatedFrom&gt;</td>
<td>39</td>
<td>39</td>
<td>100</td>
<td>95.52</td>
<td>4.48</td>
<td>26280</td>
</tr>
<tr>
<td>&lt;happenedIn&gt;</td>
<td>36</td>
<td>36</td>
<td>100</td>
<td>95.18</td>
<td>4.82</td>
<td>166496</td>
</tr>
<tr>
<td>&lt;happenedOnDate&gt;</td>
<td>43</td>
<td>43</td>
<td>100</td>
<td>95.9</td>
<td>4.1</td>
<td>182572</td>
</tr>
<tr>
<td>&lt;hasAcademicAdvisor&gt;</td>
<td>44</td>
<td>44</td>
<td>100</td>
<td>95.99</td>
<td>4.01</td>
<td>2895</td>
</tr>
<tr>
<td>&lt;hasArea&gt;</td>
<td>42</td>
<td>42</td>
<td>100</td>
<td>95.81</td>
<td>4.19</td>
<td>129909</td>
</tr>
<tr>
<td>&lt;hasBudget&gt;</td>
<td>44</td>
<td>44</td>
<td>100</td>
<td>95.99</td>
<td>4.01</td>
<td>715</td>
</tr>
<tr>
<td>&lt;hasCapital&gt;</td>
<td>63</td>
<td>61</td>
<td>96.83</td>
<td>94.13</td>
<td>4.99</td>
<td>1821</td>
</tr>
<tr>
<td>&lt;hasChild&gt;</td>
<td>64</td>
<td>62</td>
<td>96.88</td>
<td>94.22</td>
<td>4.92</td>
<td>14998</td>
</tr>
<tr>
<td>&lt;hasCurrency&gt;</td>
<td>36</td>
<td>36</td>
<td>100</td>
<td>95.18</td>
<td>4.82</td>
<td>504</td>
</tr>
</tbody>
</table>
Database Project - Goals

- Project goal: to tackle and resolve real-life DB related development issues

- Including
  - DB design
  - Query writing
  - DB programming
  - Application design
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

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
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


<table>
<thead>
<tr>
<th>TAXONOMY</th>
<th>yagoSchema</th>
<th>The domains, ranges and confidence values of the relations</th>
<th>Preview</th>
<th>Download TRL</th>
<th>Download TSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>yagoTypes</td>
<td>yagoTypes</td>
<td>All rdf type facts of YAGO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yagoTaxonomy</td>
<td>yagoTaxonomy</td>
<td>The entire YAGO taxonomy. These are all rdf:subClassOf facts derived from Wikipedia and from WordNet.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yagoTransitiveType</td>
<td>yagoTransitiveType</td>
<td>Transitive closure of all rdf:subClassOf facts</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| SIMPLETAX              | yagoSimpleTaxonomy | A simplified rdf:subClassOf taxonomy. This taxonomy contains just WordNet leaves, the main YAGO branches, and one Thing. Use with yagoSimpleTypes. |         |               |               |
| yagoSimpleTypes       | yagoSimpleTypes   | 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. |         |               |               |

| CORE                   | yagoFacts      | All facts of YAGO that hold between instances            |         |               |               |
| yagoLabels             | yagoLabels     | All facts of YAGO that contain labels (rdfs:label, skos:prefLabel, isoPreferredMeningOf, hasOwnName, hasFamilyName, hasGloss) |         |               |               |
| yagoLiteralFacts       | yagoLiteralFacts | All facts of YAGO that contain literals (except labels) |         |               |               |

| GEOAABBES              | yagoGeonamesClasIds | IDs from GeoNames classes                               |         |               |               |
| yagoGeonamesClasses   | yagoGeonamesClasses | Classes from GeoNames                                   |         |               |               |
| yagoGeonamesEntityIds | yagoGeonamesEntityIds | IDs from GeoNames entities                            |         |               |               |

| META                   | yagoStatistics | Statistics about YAGO and YAGO themes                   |         |               |               |
YAGO data – HowTo (cont.)

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

<table>
<thead>
<tr>
<th>Format:</th>
<th>yago-id</th>
<th>entity</th>
<th>relation</th>
<th>entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id_zik11d_88c_ehg9uq&gt;</td>
<td>&lt;A&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory_Vowel_letters&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;id_zik11d_88c_w3c6wm&gt;</td>
<td>&lt;A&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory_ISO_basic_Latin_letters&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;id_1bsrlah_88c_1s6g79w&gt;</td>
<td>&lt;Alabama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory_States_of_the_United_States&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;id_3ienox_88c_4retae&gt;</td>
<td>&lt;Achilles&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory_People_of_the_Trojan_War&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;id_3ienox_88c_1rk49a2&gt;</td>
<td>&lt;Achilles&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory_Pederastic_heroes_and_deities&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;id_3ienox_88c_s57m6o&gt;</td>
<td>&lt;Achilles&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory_Kings_of_the_Myrmidons&gt;</td>
<td></td>
</tr>
</tbody>
</table>

- 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](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

```xml
<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_4retaee> <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 data - Core

- **yagoFacts** – facts between instances
  + A complete list of relations – in Taxonomy, 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
    ```
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.
Example

- 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
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.,

\[
\text{<Lionel\_Messi>} \quad \text{rdf:type} \quad \text{<wordnet\_player\_110439851>}
\]
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

```xml
<Lionel_Messi> skos:prefLabel "Lionel Messi"@eng
```

+ Birthdate and height – from yagoLiteralFacts

```xml
<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 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

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 by:
- Search by album ID:
- Search by other parameters:
  - Album name:
  - Artist: Led Zeppelin
  - Year from: To:
  - Song name(s): Dazed and Confused
- Genre(s):
  - Rock
  - Jazz
  - Other:
- Stock: All

Search Results:

<table>
<thead>
<tr>
<th>Album ID</th>
<th>Album Name</th>
<th>Artist</th>
<th>Year</th>
<th>Genre</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>882849</td>
<td>electric magic o</td>
<td>led zeppelin</td>
<td>0</td>
<td>rock &amp; roll</td>
<td>78:44</td>
</tr>
<tr>
<td>911119</td>
<td>dazed and con...</td>
<td>led zeppelin</td>
<td>0</td>
<td>rock &amp; roll</td>
<td>56:04</td>
</tr>
<tr>
<td>1264973</td>
<td>whole lotta fo...</td>
<td>led zeppelin</td>
<td>1969</td>
<td>classic ro...</td>
<td>75:09</td>
</tr>
<tr>
<td>1151298</td>
<td>the sex machi...</td>
<td>led zeppelin</td>
<td>1975</td>
<td>rock &amp; roll</td>
<td>67:11</td>
</tr>
</tbody>
</table>

Show Song List:

<table>
<thead>
<tr>
<th>Track</th>
<th>Song name</th>
<th>Artist</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sweet baby</td>
<td>Various Artists</td>
<td>2:59</td>
</tr>
<tr>
<td>2</td>
<td>I can’t quit y...</td>
<td>Various Artists</td>
<td>6:50</td>
</tr>
<tr>
<td>3</td>
<td>dazed and conf...</td>
<td>Various Artists</td>
<td>14:59</td>
</tr>
<tr>
<td>4</td>
<td>you shock me</td>
<td>Various Artists</td>
<td>10:42</td>
</tr>
<tr>
<td>5</td>
<td>how many mor...</td>
<td>Various Artists</td>
<td>10:45</td>
</tr>
<tr>
<td>6</td>
<td>bye bye baby</td>
<td>Various Artists</td>
<td>11:42</td>
</tr>
</tbody>
</table>
Past years projects
Past years projects

[Image of MusicTube interface with options]

[Image of MusicTube Disc Search interface]

[Table showing song information]
Past years projects
Past years projects

![Music Marketplace interface](image.png)

- **My Requests**:
  - **Album Title**: Luftslottet som sprengtes
  - **New**: 
  - **Album Title**: Rafi Resurrected
  - **New**: 
  - **Album Title**: Qurbani (remastered)

- **My Offers**:
  - **Album Title**: Saturday Night Fever
  - **New**: 
  - **Album Title**: We Are... Sniffing Glue ....

- **Users**:
  - **Username**: guy
  - **Email**: guy.ba...
  - **Can give you**: Qurbani (remastered)
  - **Wants from you**: Saturday Night Fever
  - **Match**: !
  - **Username**: guy
  - **Email**: guy.ba...
  - **Can give you**: Luftslottet som sprengtes
  - **Wants from you**: Saturday Night Fever
  - **Match**: !
  - **Username**: guy
  - **Email**: guy.ba...
  - **Can give you**: Rafi Resurrected
  - **Wants from you**: Saturday Night Fever
  - **Match**: !
  - **Username**: guy
  - **Email**: guy.ba...
  - **Can give you**: Rafi Resurrected
  - **Wants from you**: Luftslottet som sprengtes
  - **Match**: !
  - **Username**: guy
  - **Email**: guy.ba...
  - **Can give you**: Qurbani (remastered)
  - **Wants from you**: Saturday Night Fever
  - **Match**: !
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
(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\textsuperscript{th} – Project distribution

April 18\textsuperscript{th} – Last date for submitting the team member names

May 21\textsuperscript{st} – “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\textsuperscript{th} – Project due!
+ Aim to submit a week before, to avoid network crushes, mysterious illnesses...
DB Project

בהצלחה!