DB Project

Database Systems

Spring 2013

Database project – YAGO

WIKIPEDIA
The Free Encyclopedia

WordNet
A lexical database for English

GeoNames

Yet **A**nother **G**reat **O**ntology

yago
select knowledge

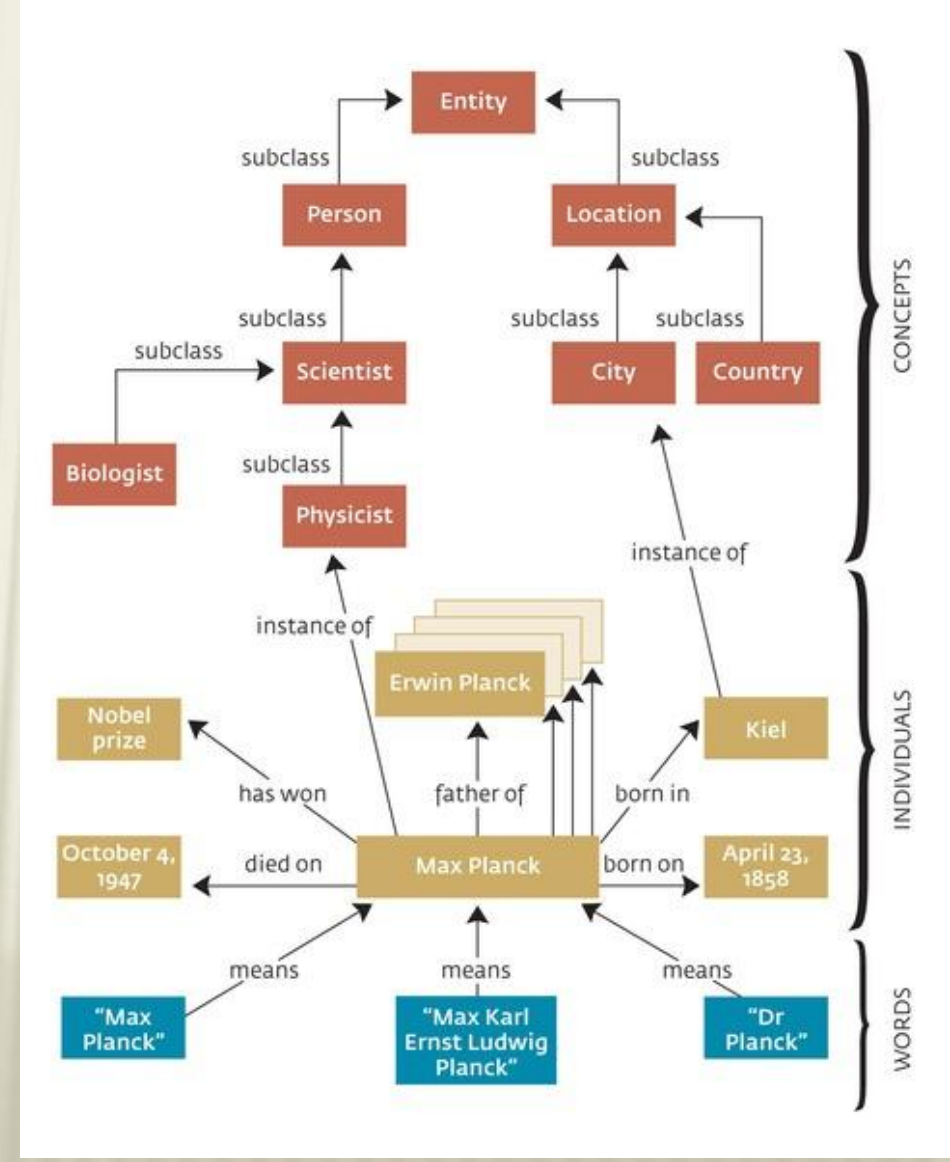
About YAGO*

- 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

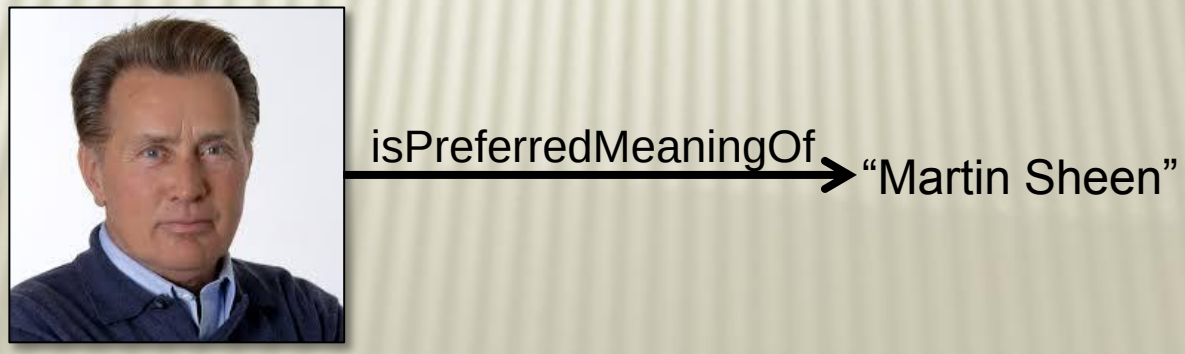
- A Taxonomy of concept classes
- At the bottom – instances, facts about instances
  - This is typically the interesting part



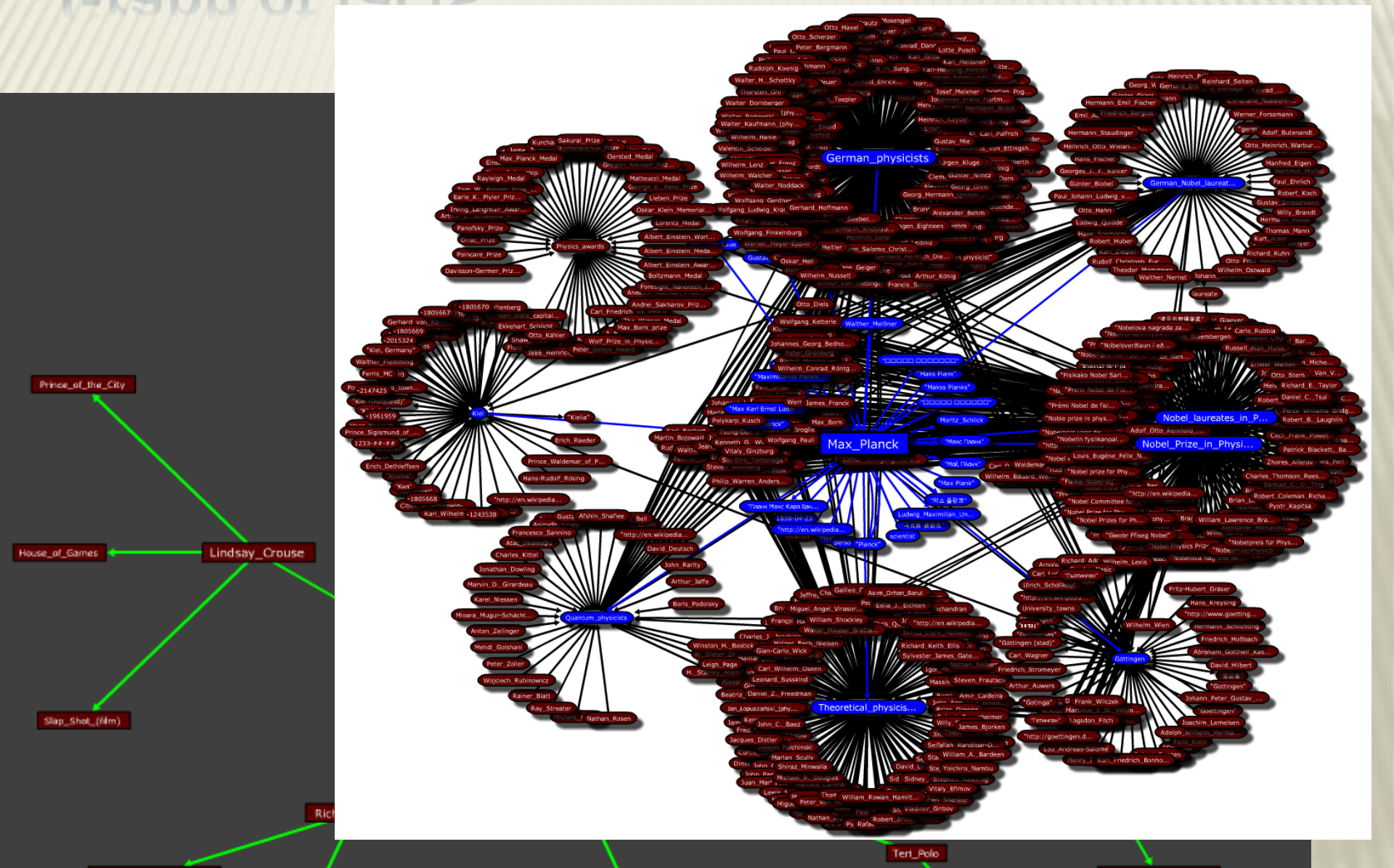
A fact

- A particular relationship that holds between two instances

- Or, an instance and a literal (string)



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

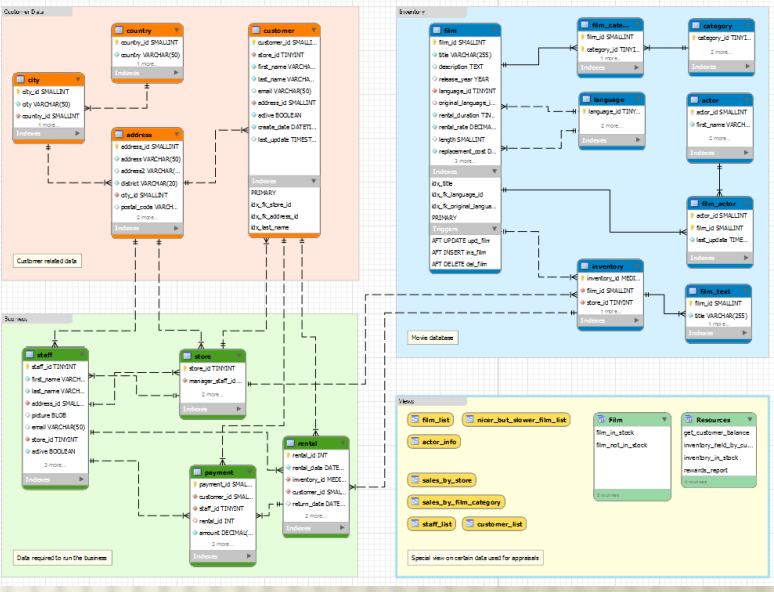
Database Project - Requirements

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

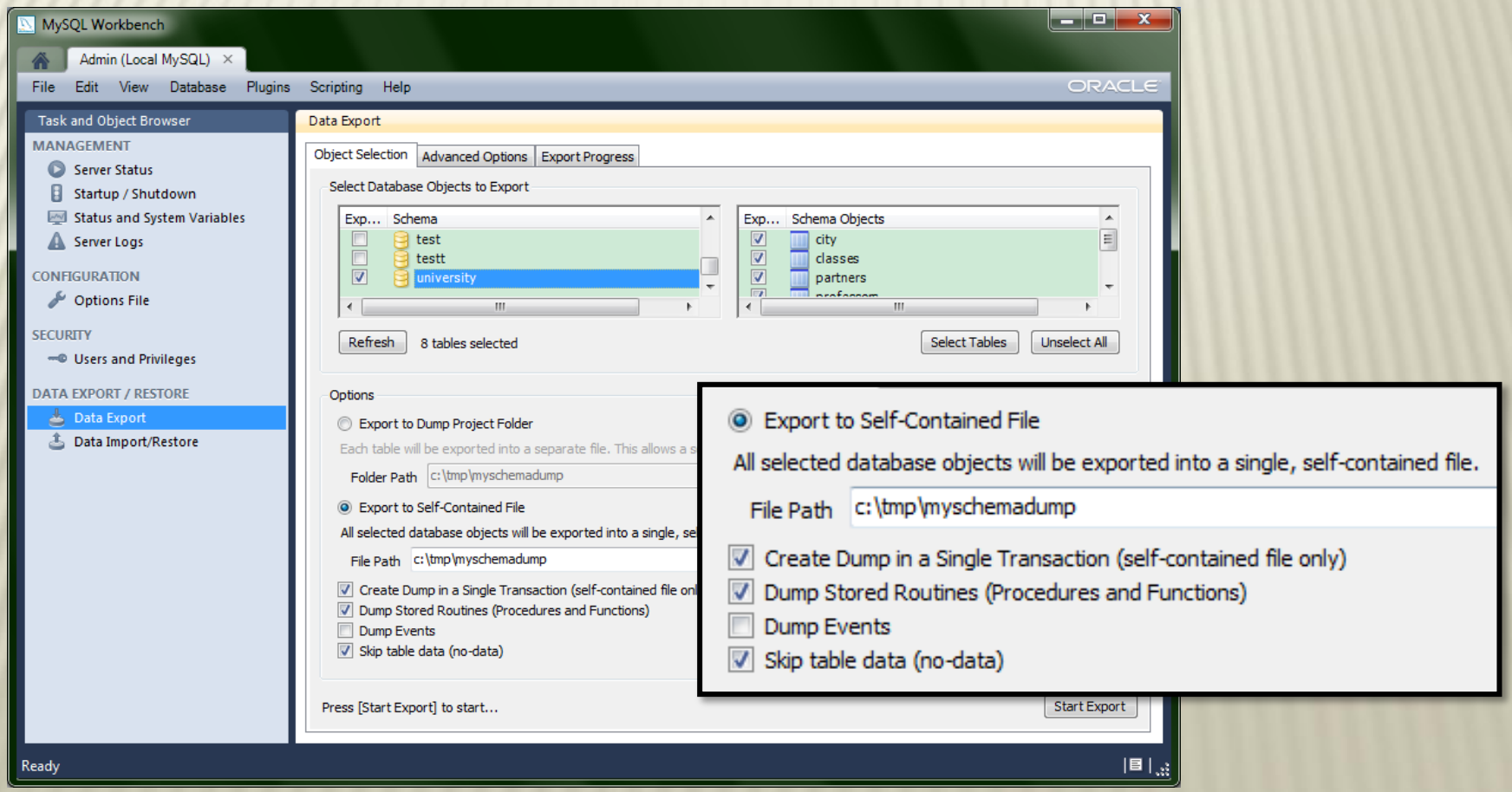


- Tables, indexes, keys and foreign keys
- Avoid redundant information
- Allow efficient queries
- The script for generating the schema should be submitted with the project

- 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

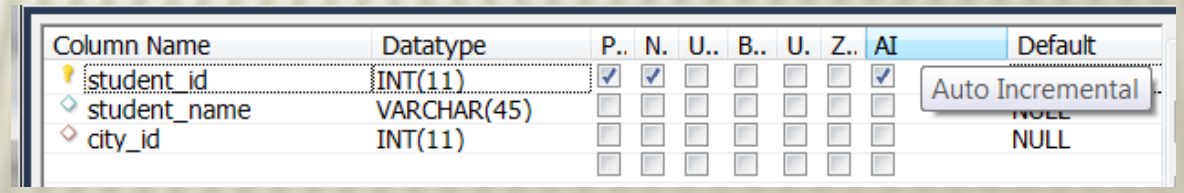


3. Load data from YAGO

- 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



5. Write an application

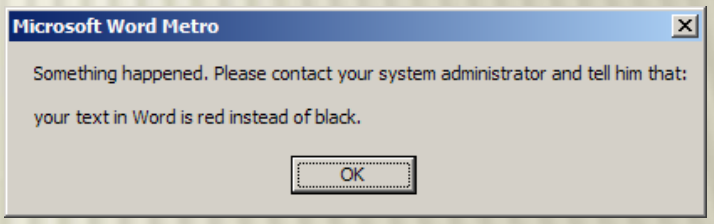
- In java, using JDBC
- Desktop application
- SWT for GUI (other open-source packages such as Swing, Qt Jambi…)
- Any other open-source packages, *except hibernate and similar packages*
- According to DB programming principles
- **Important:** separate the code of the UI, the core logic and the DB

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

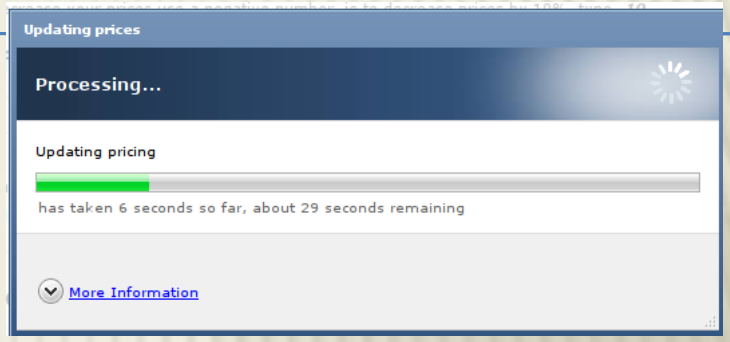
5. Write an application (cont.)

- Should be usable and easy to understand
- Should be fault tolerant
  - Every exception should be caught, and a user-friendly message should be displayed

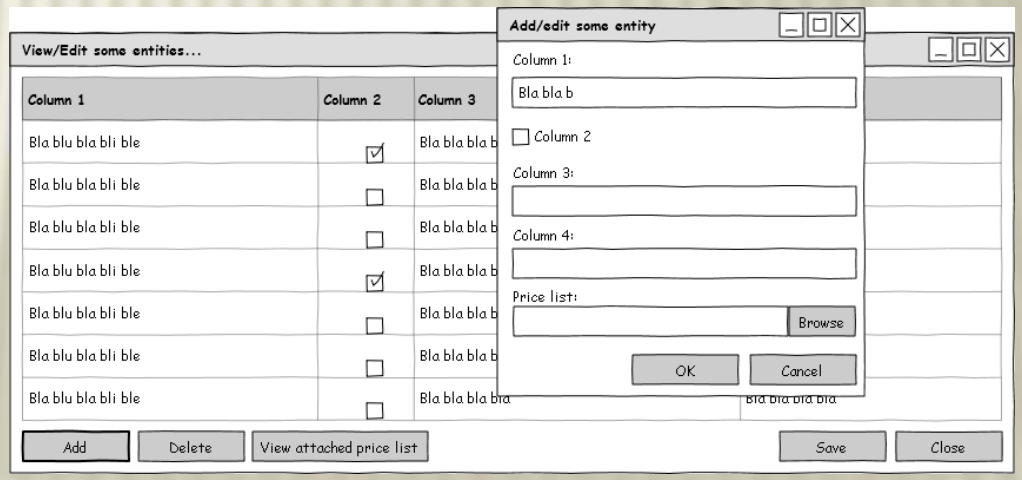


- Test your application
  - Install on different environments
  - Portable:
    - Copy-paste, create DB schema, edit configuration and… play!

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

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>

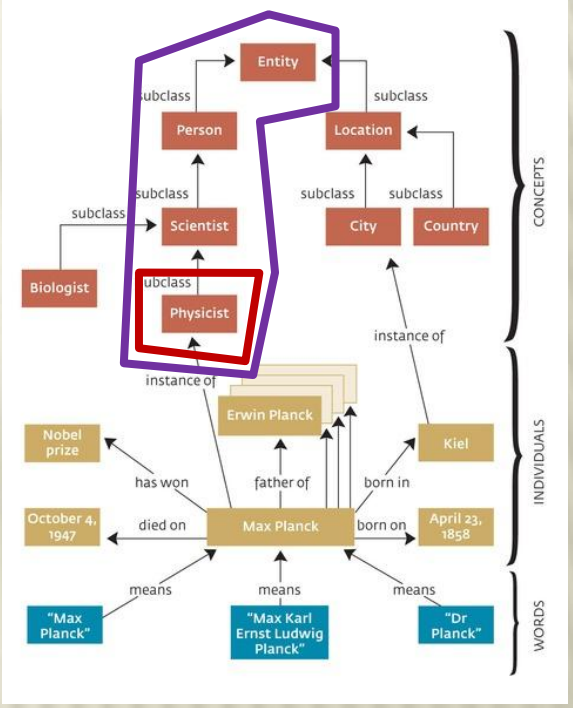
- YAGO entities and relations are marked by < > (e.g., <Achilles>)
- Others are taken from rdf, rdfs, owl, skos… (e.g., rdf:type)
- Literals are marked by " "
  - Strings with optional locale, e.g., "Big tent"@eng
  - Others with datatype, e.g., "1977-08-16"^^xsd:date, "70"^<m>
- See also: http://www.mpi-inf.mpg.de/yago-naga/yago/faq.html

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 entitites, 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>	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 data - Core

- yagoFacts – facts between instances
  - A complete list of relations – in Taxomony, yagoSchema

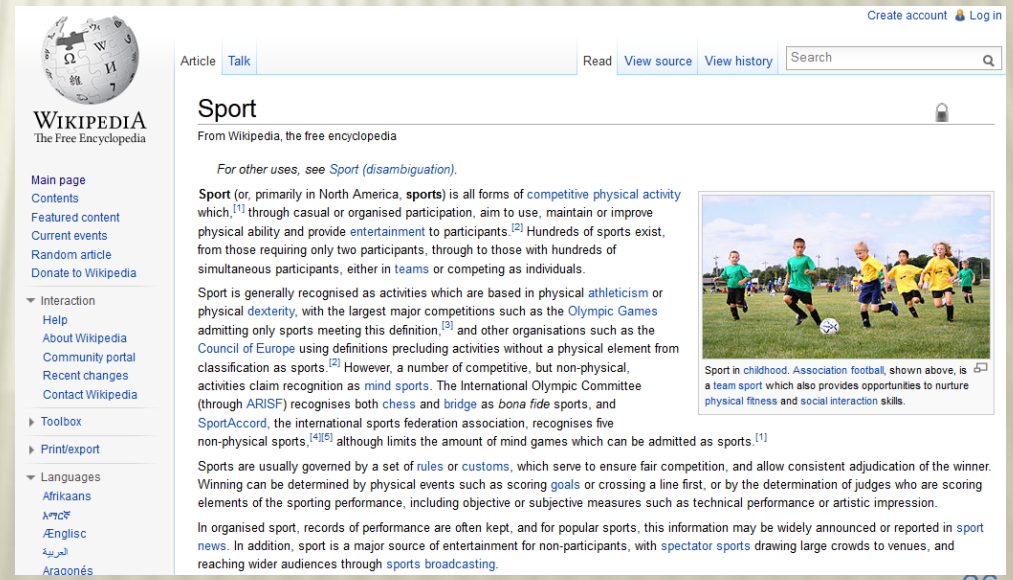
  <Martin_Sheen> <hasChild> <Charlie_Sheen>
- YagoLabels – names of entities.
  - There may be many labels! use skos:prefLabel

  <Martin_Sheen> skos:prefLabel "Martin Sheen"@eng
- yagoLiteralFacts –other facts with literals
  - Often properties of the entity

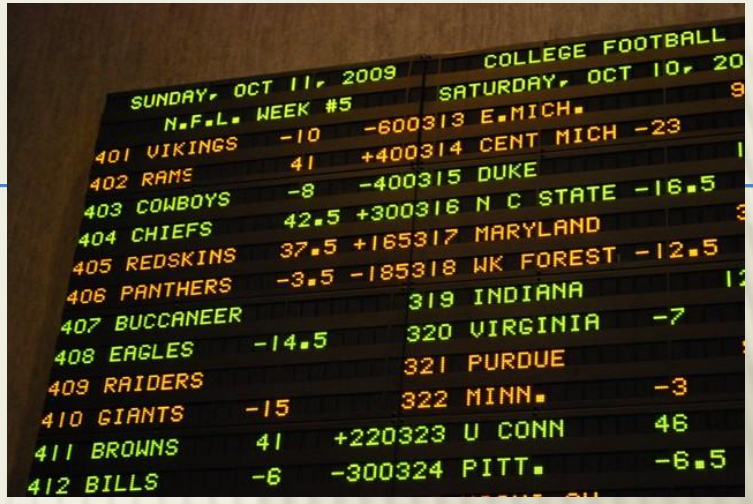
  <Martin_Sheen> <wasBornOnDate> "1940-08-03"^^xsd:date

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.



Example



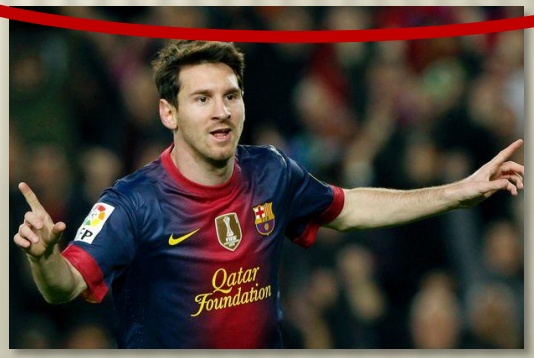
- Editing capabilities for YAGO data: add/remove/edit all players, teams, games…
- Data of your own: odds, bets…
- Your tables:
  - Players, Teams, Users, Bets
  - Linking tables: Player_team, User_bets

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

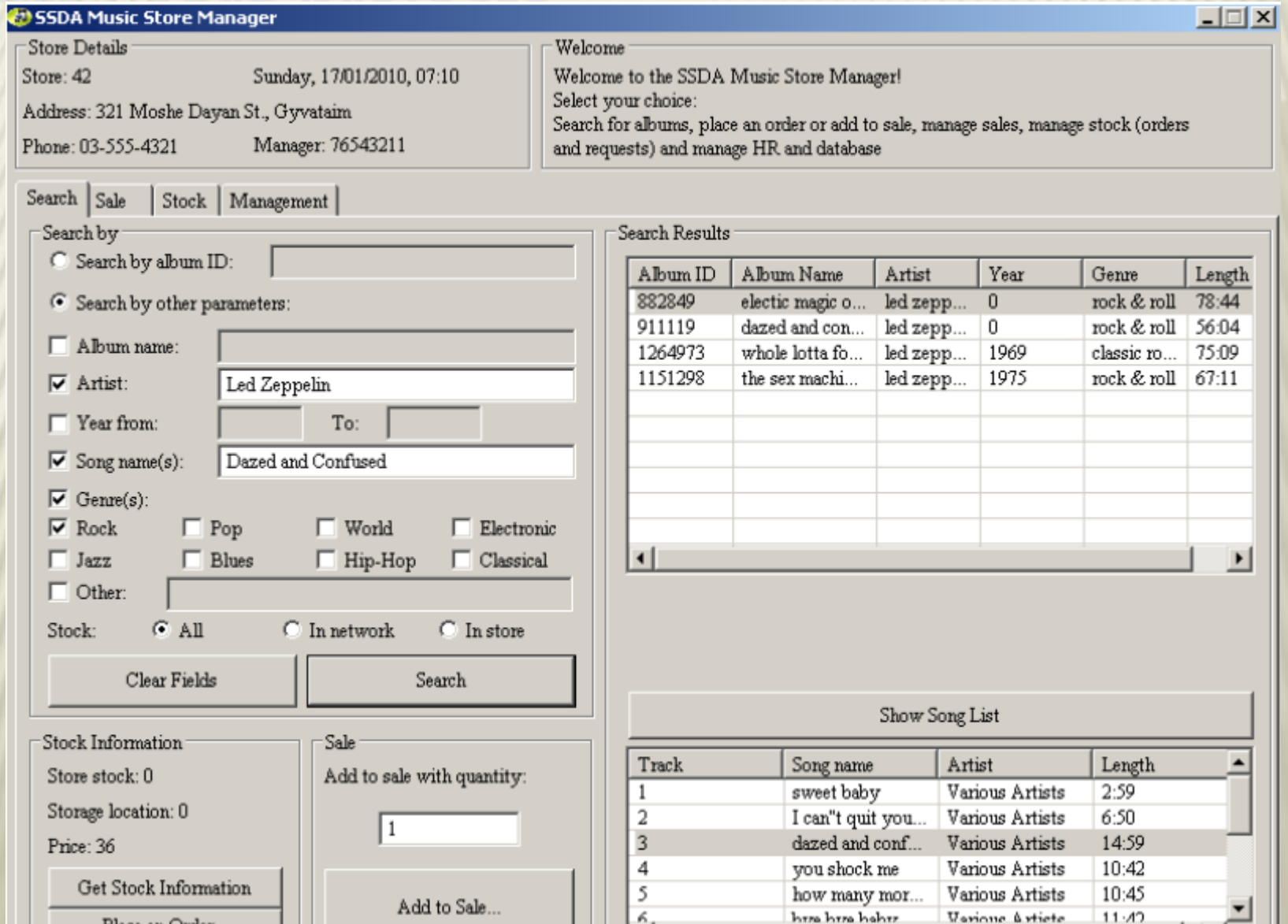
- What do we do when a value is missing?
- What do we do when the data in invalid?
- What do we do when there is more than one value?

<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

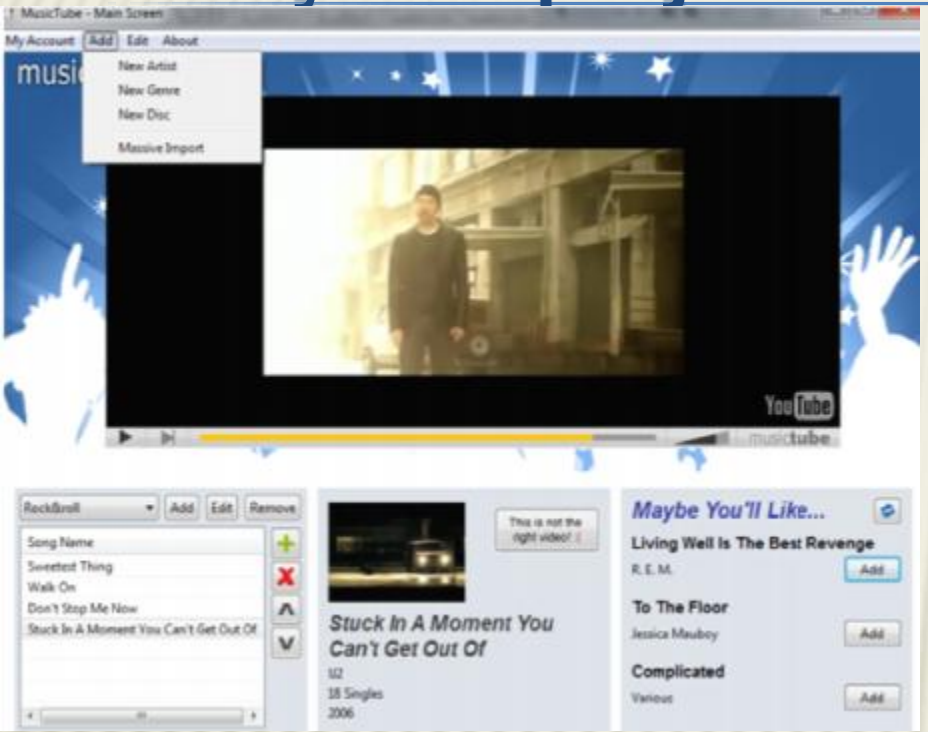
- You do not have to fix errors in YAGO's data (but you can allow the application users to do so)
- You can choose an arbitrary value if there are many (where this makes sense! playsFor can be many-to-one, actedIn cannot)
- You can use an additional data source to complete missing data (must be freely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

Past years projects

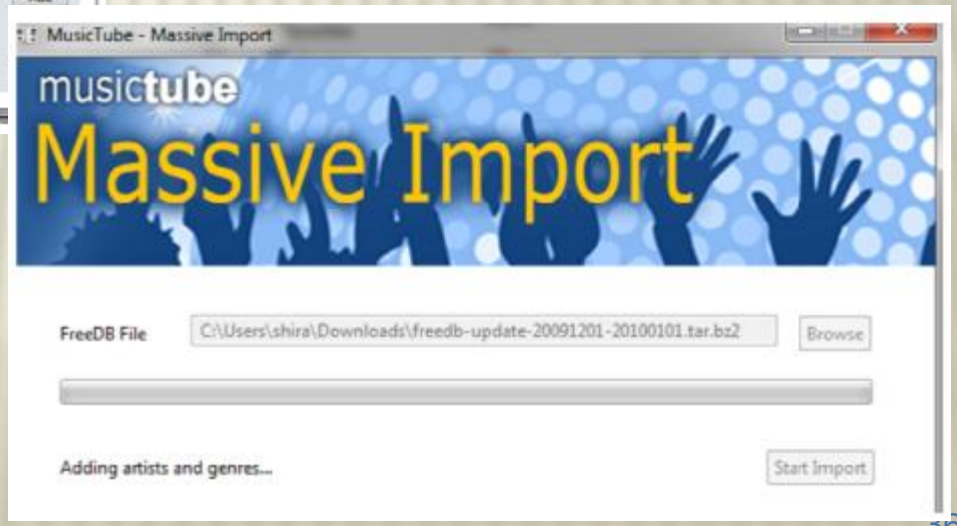


Past years projects

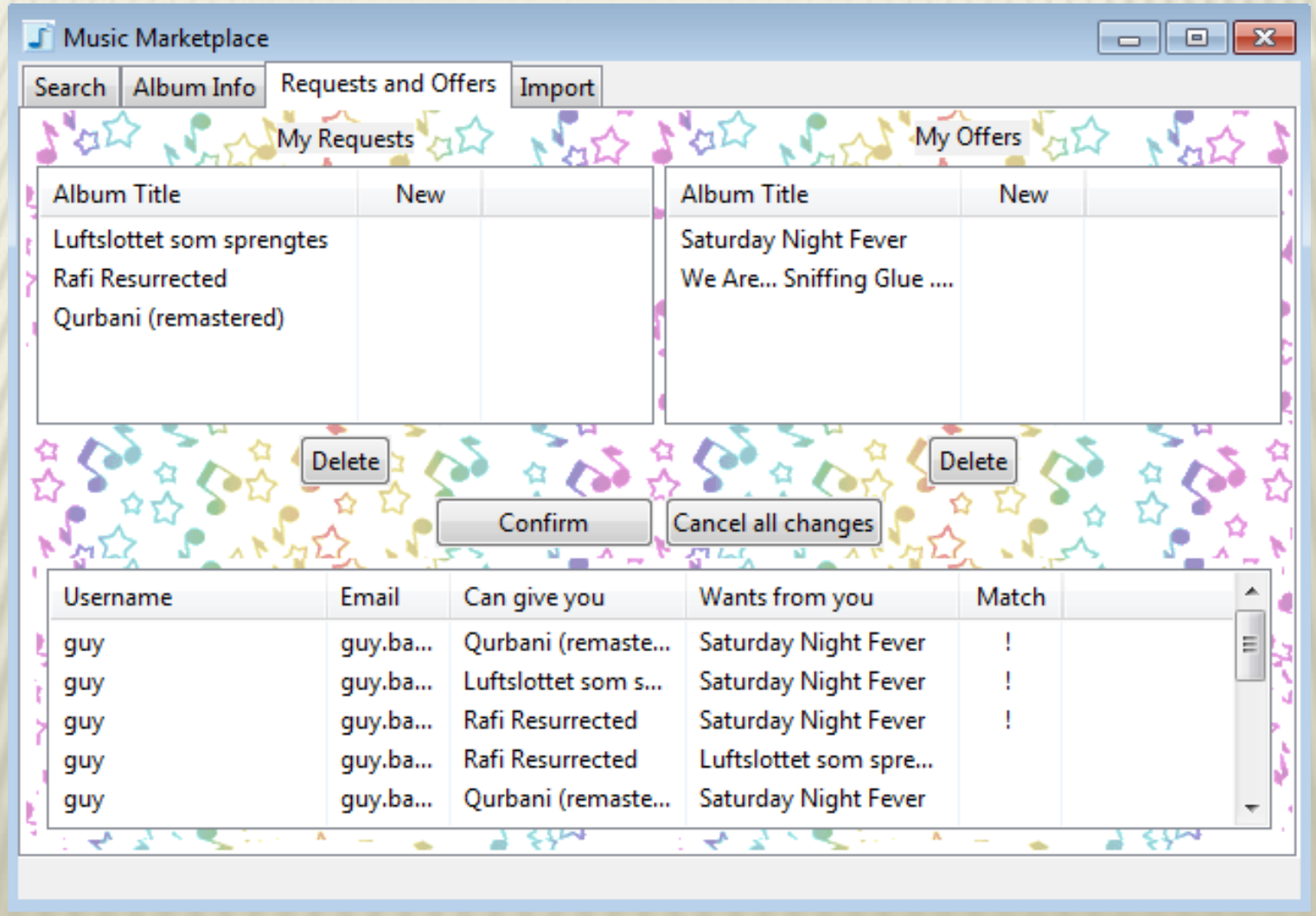
Past years projects



Past years projects



Past years projects



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 9th** – Project distribution
- **April 18th** – Last date for submitting the team member names
- **May 21st** – "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 18th** – Project due!
  - Aim to submit a week before, to avoid network crushes, mysterious illnesses…

DB Project

בהצלחה!