DB Project

Database Systems

Spring 2015

Database project – YAGO

About YAGO*

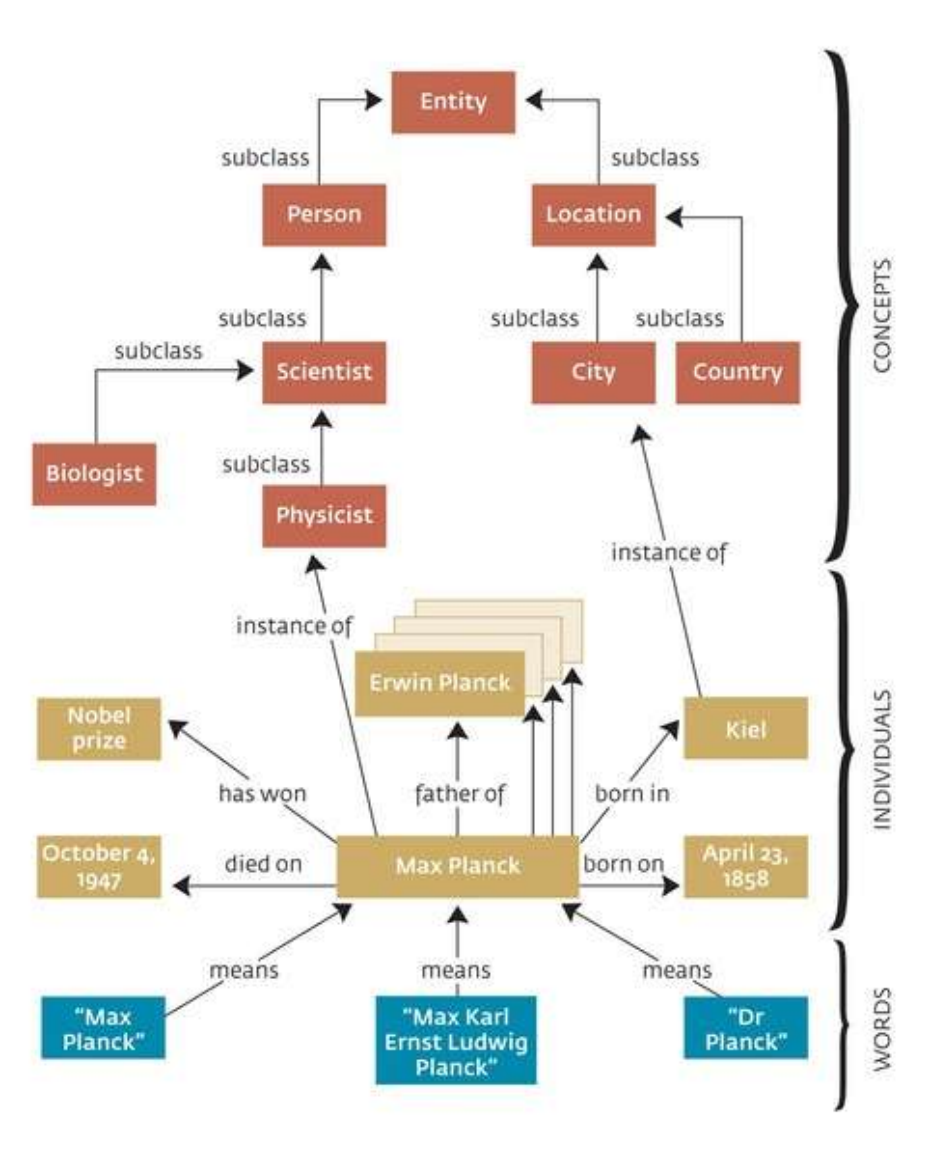
- A huge semantic knowledge base
  - knowledge base – a special kind of DB for knowledge management (e.g., facts)
  - Semantic – related to the Semantic Web where presentation is assigned a meaning http://www.youtube.com/watch?v=TJfrNo3Z-DU&feature=player_embedded

yago
select knowledge

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

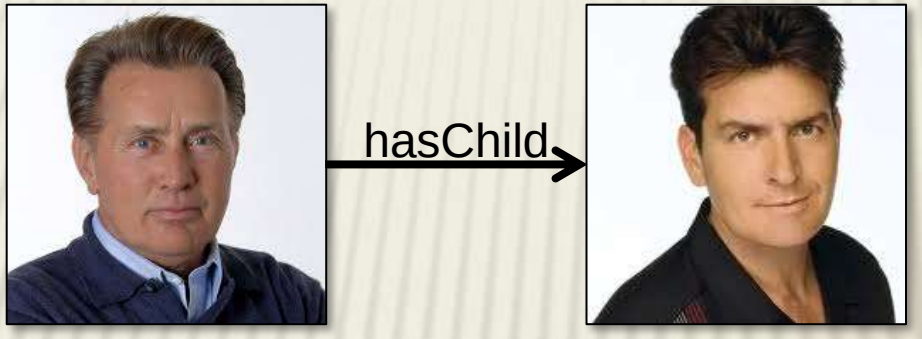
YAGO is an Ontology

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

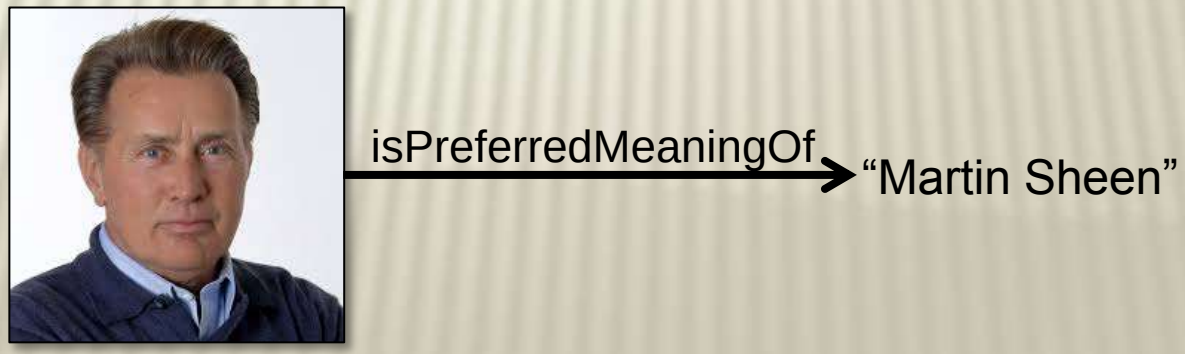


A fact

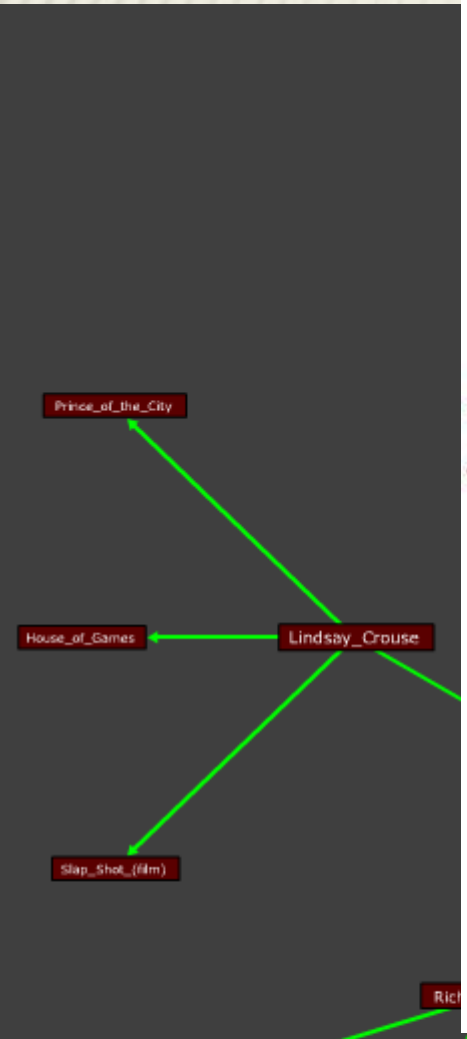
- A particular relationship that holds between two instances

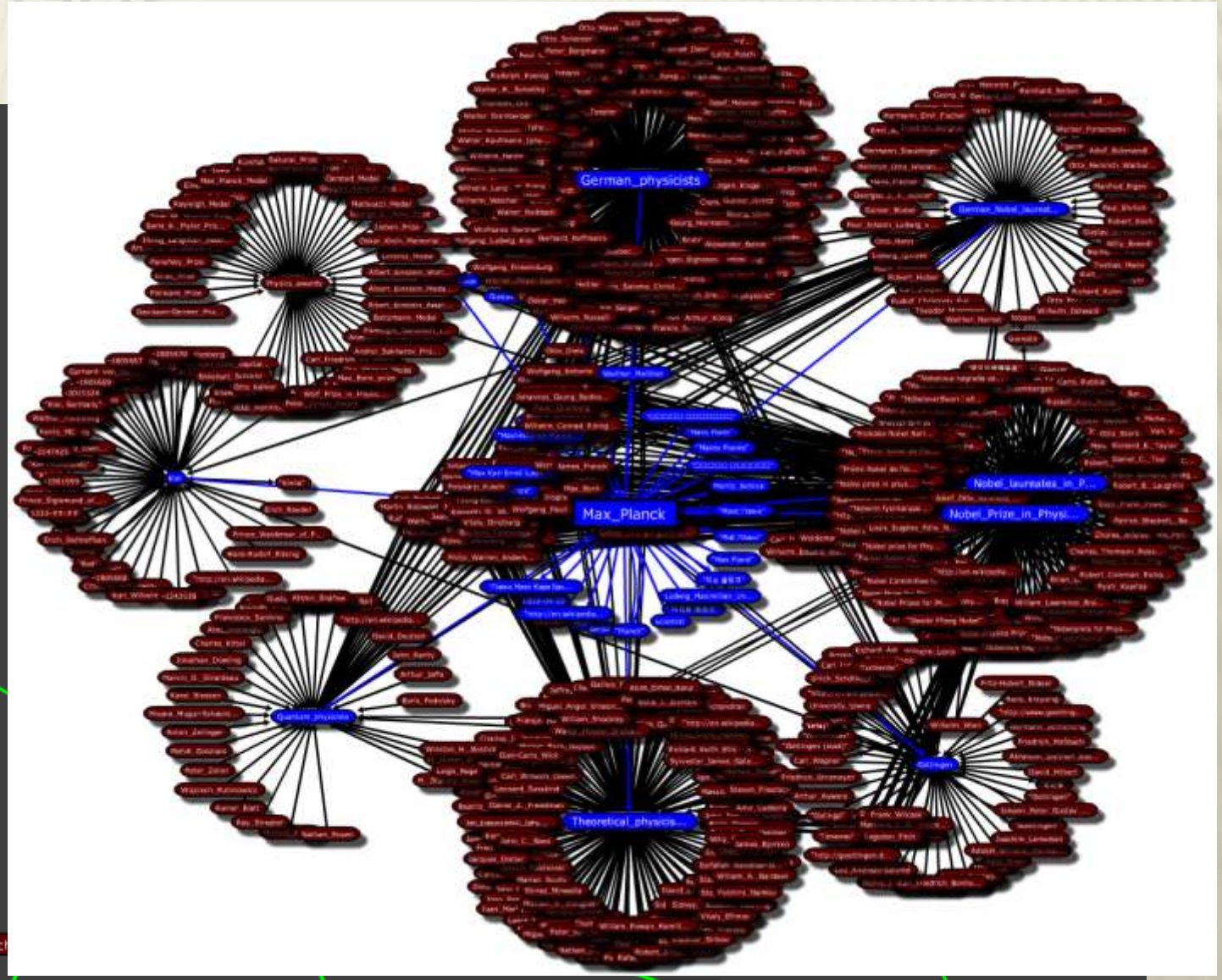


- Or, an instance and a literal (string)



Graph of facts





More about YAGO

- More than 10 million entities
- More than 120 million facts
- High (but not perfect!) accuracy
- Links 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

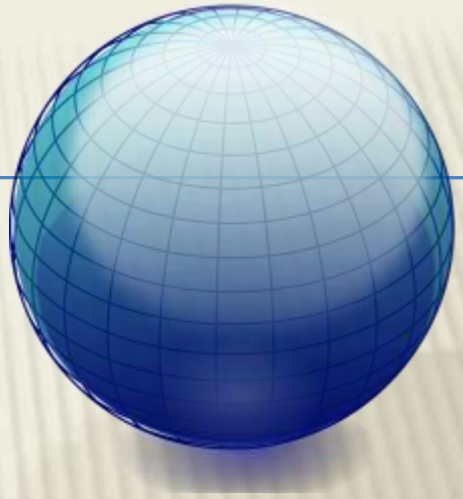
- Project goal: to tackle and resolve real-life DB related development issues

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

The Theme of the Projects

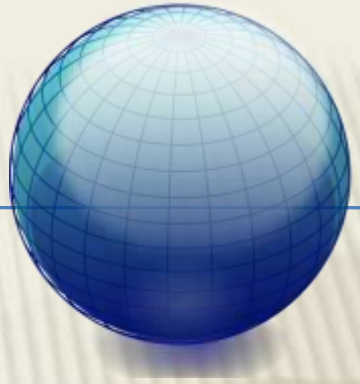
Around the Globe

Around the Globe



- Applications that involve locations, countries, and languages
- A lot of data in YAGO **has locations** (birth places, historic events, sights, movies, books, sports teams…)
- A lot of data in YAGO **is about locations** (e.g., NYC is in New York state which is in the US, properties of cities and countries, GeoNames data)
- YAGO Data partly comes from Wikipedia and Wordnet **in different lanuages**

What will Your App Do with this?



Whatever you want!

- Games that involve locations (e.g., trivia, puzzles, word games)
- Social applications that display locations on a map
- Search engines that use geographical criteria
- Applications related to travel recommendations or reviews
- Something even better?...

... But make sure that a single grader can also check it!

Where to Find the Data?

- Yago facts about

<diedIn>, <exports>, <happenedIn>, <hasCapital>, <hasCurrency>, <hasOfficialLanguage>, <imports>, <isCitizenOf>, <isLeaderOf>, <isLocatedIn>, <isPoliticianOf>, <livesIn>, <wasBornIn>, <isConnectedTo>, <byTransport> (meta-fact)

(but you can use, in addition, any other data in YAGO)

- GeoNames data (in YAGO downloads)
- YAGO multilingual labels (in YAGO downloads)
- YAGO extracts data from multilingual Wikipedia Infoboxes (in YAGO downloads)
- You can also combine data from external sources, e.g., wikidata, LinkedGeoData, Foursquare – as long as the core of your data is from YAGO

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

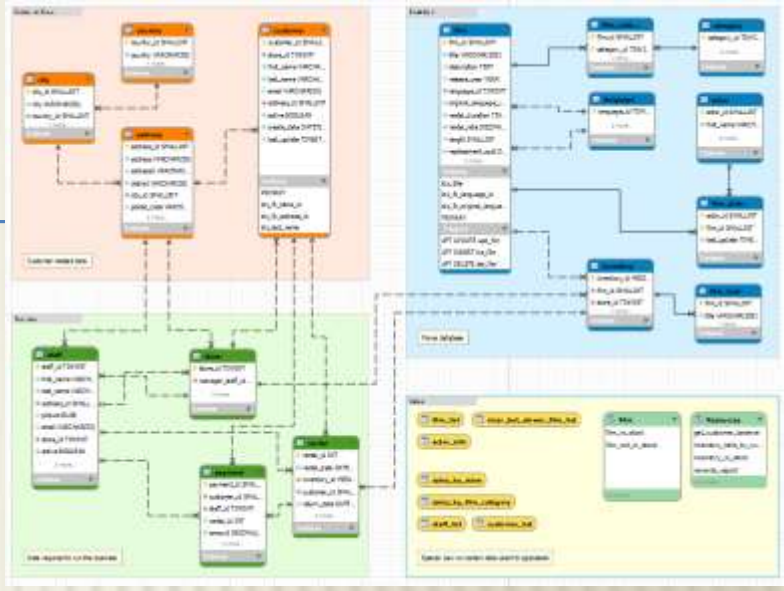
Parts of your application!

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, in light of the theme
  - YOU should want to use it…
  - **Tip:** first inspect the available data
  - **Tip:** must-have and nice-to-have features
  - Can be interesting even if the UI is simple

2. Design a DB Schema

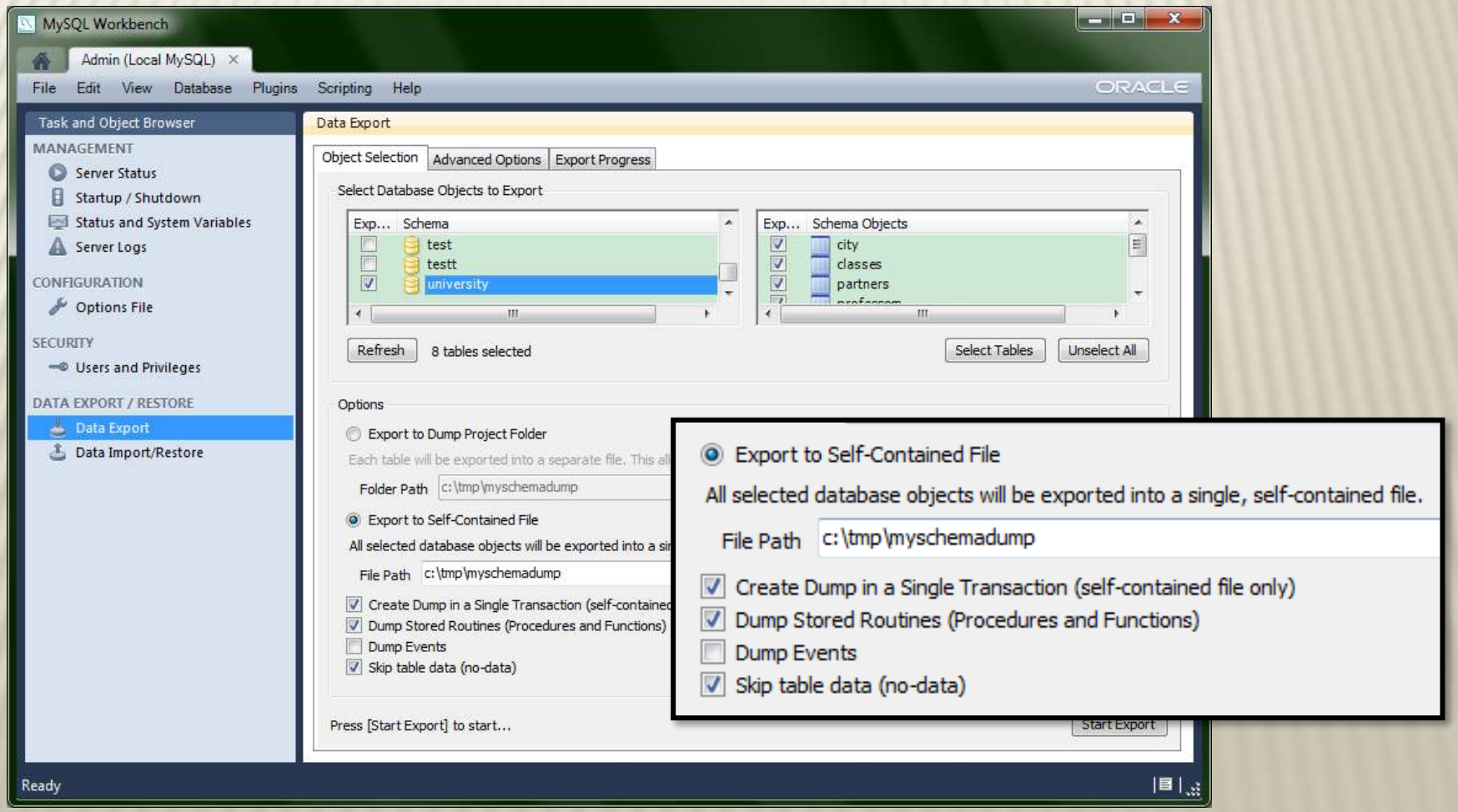


- Tables, indexes, keys and foreign keys
- Avoid redundant information
- Allow efficient queries
- The script for generating the schema should be submitted with the project

- More about design in the following lectures

2. Creating the SQL Script

- http://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html

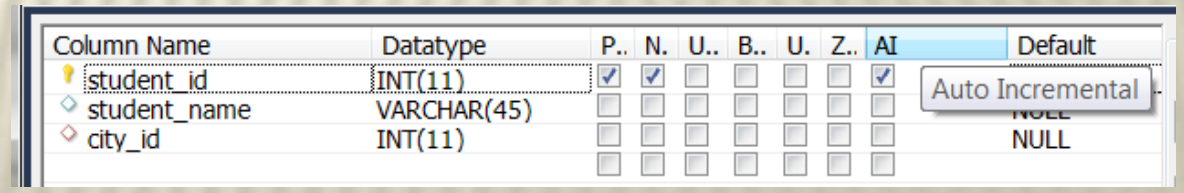


3. Load data from YAGO

- The entire database of YAGO is available online
- Choose the files you need
  - They will be saved locally in the file system
- In your java code, read the relevant data
  - YAGO is huge, and may not fit into your quota
  - keep data in Sets/Lists/Maps/arrays/custom types
- This process may be done in several batches
- HowTo below

4. Update the DB

- 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?)
  - 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

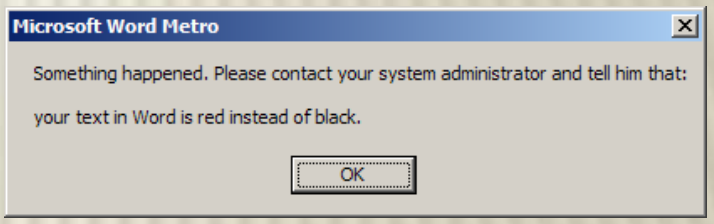


5. Write an application

- 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

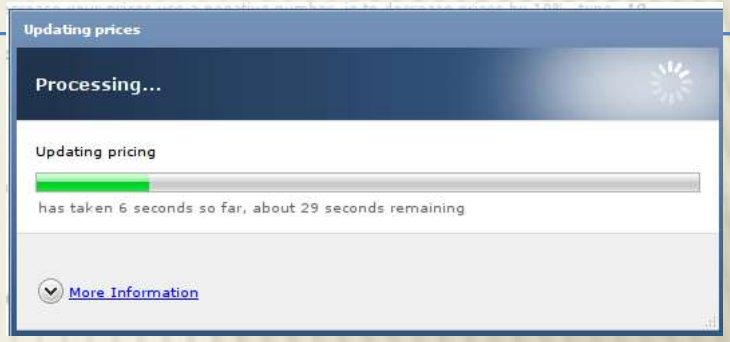
5. Write an application (cont.)

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

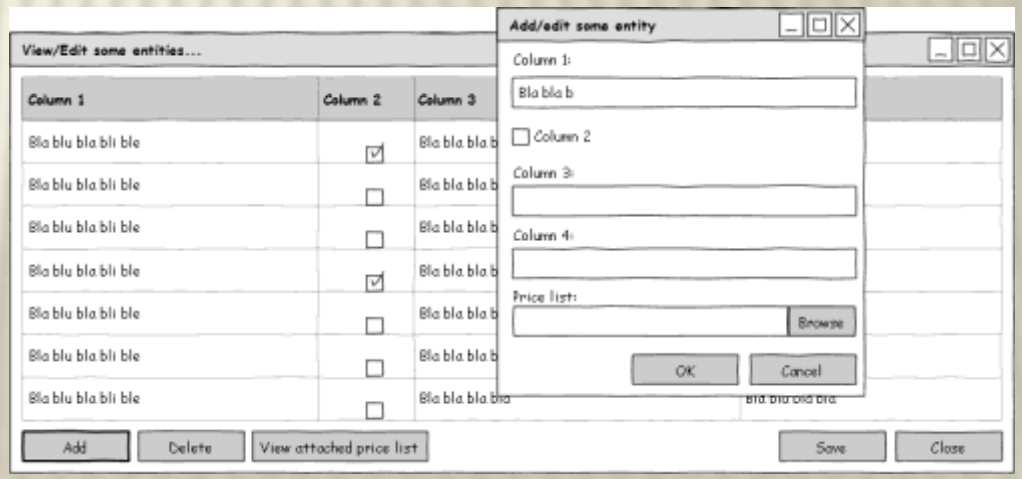


- Test your application
  - Install on different environments
  - Portable:
    - Copy-paste, create DB schema, edit configuration and… play!

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



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: fact-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>

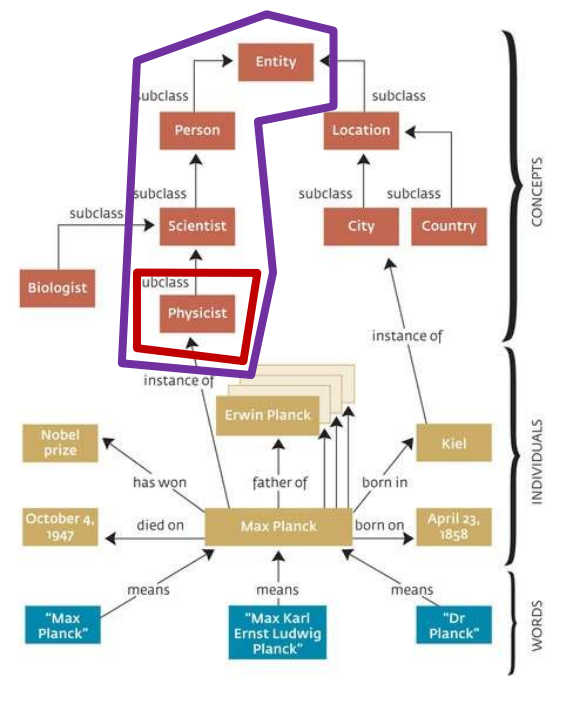
- YAGO entities and relations are marked by < > (e.g., <Achilles>)
- Others are taken from rdf, rdfs, owl, skos… (e.g., rdf:type)
- Literals are marked by " "
  - Strings with optional locale, e.g., "Big tent"@eng
  - Others with datatype, e.g., "1977-08-16"^^xsd:date, "70"^<m>
- See also: http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/r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

YAGO data – HowTo (cont.)

- You can also download just the portions of YAGO that you need. Some examples for useful downloads:
  - YagoTransitiveType: each entity in YAGO with all of its classes
  - YagoTaxonomy: The hierarchy of classes
  - YagoLiteralFacts/DateFacts: facts of the form entity-relation-literal/date
  - YagoLabels: The names of entities as strings
  - YagoFacts: facts if the form entity-relation-entity\
  - yagoGeonamesClasses: similar to taxonomy but with geographical places
  - yagoGeonamesOnlyData: similar to YagoFacts with entities from GeoNames
  - YagoGeonamesType: similar to YagoType with GeoNames entities
  - YagoMetaFacts: facts about facts (e.g., when did a certain fact occur)
  - MULTILINGUAL: The multilingual names for entities.

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

- yagoDateFacts

  <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

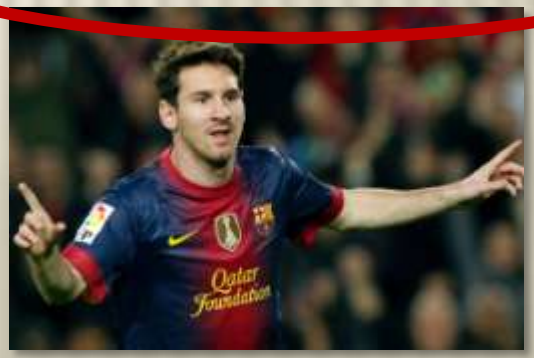
- Editing capabilities for YAGO data: add/remove/edit all players, teams, games…
- Data of your own: odds, bets…
- Your tables:
  - Players, Teams, Users, Bets
  - Linking tables: Player_team, User_bets

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/DateFacts

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

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 must be many-to-many)
- 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: Led Zeppelin
- Year from: To:
- Song name(s): Dazed and Confused
- Genre(s):
- Rock, Pop, World, Electronic
- Jazz, Blues, Hip-Hop, Classical
- Other:

Stock: All, In network, In store

Clear Fields | Search

Search Results

Album ID	Album Name	Artist	Year	Genre	Length
882849	electic 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	bye bye baby...	Various Artists	11:42

Stock Information
Store stock: 0
Storage location: 0
Price: 36

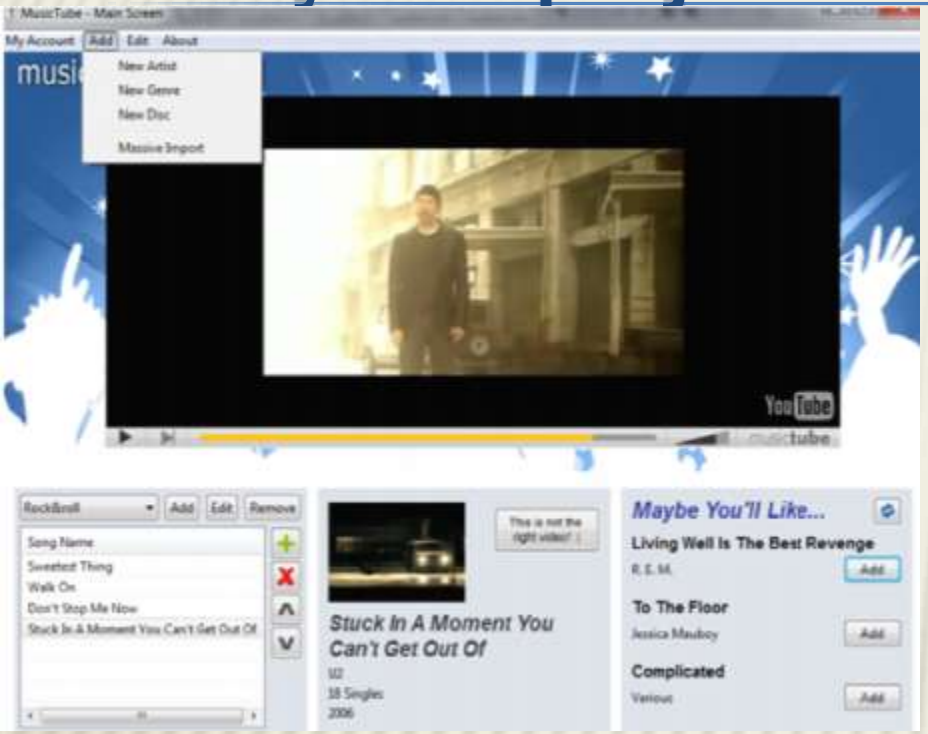
Get Stock Information | Place an Order

Sale
Add to sale with quantity: 1

Add to Sale...

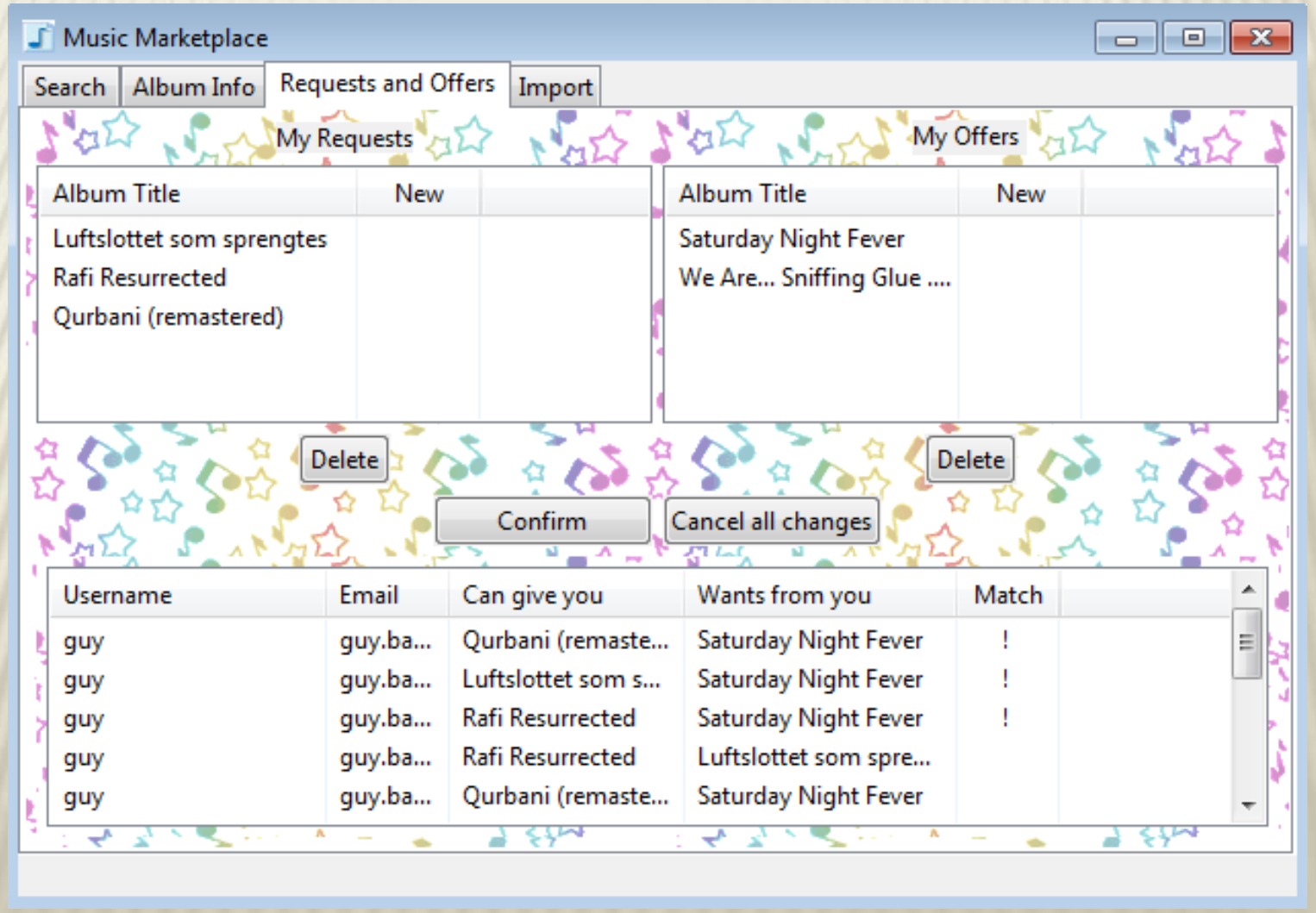
Past years projects

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 editing of the data (add/update/remove) – e.g., artists/categories/values…
- Configuration – for DB connection, OS, etc.

Database Project - Bureaucracy

- Work in groups of 4
- Submission database is MySQL in TAU
- Java, SWT (or other UI package)
- 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/databases201415/assignments/

Time schedule

- April 28th – Project distribution
- May 5th – Last date for submitting the team member names
- May 24th – submit via moodle a pdf document (2 A4 pages)with
  - Short description of the application idea
  - DB design
  - Work plan - what is left to do, who does what and when
- May 26th – "Project days"
  - I will meet (very briefly) with each team
  - Data should already be uploaded to the school DB
  - Optional – a presentation, a demonstration of the tool
  - You will be able to ask questions and get advice
- June 21st – Project due!
  - Aim to submit a week before, to avoid network problem, server crushes…

DB Project

בהצלחה!

Good Luck!