Database Specification

Databases and Web Applications Laboratory (LBAW)
Bachelor in Informatics Engineering and Computation (L.EIC)

Sérgio Nunes
Dept. Informatics Engineering
FEUP · U.Porto

Outline

- ➔ Requirements Specification (ER) delivery
- ➔ Database Specification (EBD) development
  - ➔ Conceptual Data Modeling
  - ➔ Relational Schema
  - ➔ Schema Validation and Refinement
- ➔ PostgreSQL

LBAW Plan

- ➔ Plan: https://web.fe.up.pt/~ssn/wiki/teach/lbaw
  - ➔ 4th week of classes;
  - ➔ Delivery of first component (ER);
  - ➔ Lab classes: start new artefact (EBD); work on the conceptual data model (A4).
- ➔ Groups: https://moodle.up.pt/mod/choicegroup/view.php?id=35701
  - ➔ No pending situations.
- ➔ Monitor sessions: start this Wednesday, at 15h, online
  - ➔ First session on Git and GitFlow.

Requirement Specification (ER) Delivery

Requirements Specification (ER) Delivery

- First component delivery this week (8th — 12th Nov)
- Deadline is on the day before the lab class, before 12h00. Submission steps:
- 1. **Fill the group spreadsheet checklist:**
  - update the “Group Self-Evaluation” tab
  - fill the ER, A1, A2, and A3 tabs
- 2. Verify that the component on the group’s **GitLab wiki is updated with the ER** component.
- 3. **Export the component wiki page to PDF and submit it on Moodle:**
  - Only one submission per group is necessary.
  - Ensure all images were correctly exported.
  - Only the information included in the PDF will be considered for evaluation.

Questions about ER component submission?

Database Specification (EBD) Development

Database Specification (EBD) Component

- The EBD component groups the artefacts to be made by the development team in order to support the storage and retrieval requirements identified in the requirements specification.
- It consists of three artefacts:
  - A4: Conceptual Data Model
  - A5: Relational Schema, Validation and Schema Refinement
  - A6: Indexes, Triggers, Transactions and Database Population
- https://web.fe.up.pt/~ssn/wiki/teach/lbaw/202122/artefacts

A4. Conceptual Data Model

- ➔ In this artefact the data requirements of the system are detailed.
- ➔ The Conceptual Domain Model contains the identification and description of the entities of the domain and the relationships between them.
- ➔ The Conceptual Domain Model is simplified to include only concepts (entities and relationships) of the domain that are stored in the database.
- ➔ The Conceptual Data Model is obtained by using a UML class diagram containing the classes, associations, multiplicity and roles.
- ➔ For each class, the attributes, associations and constraints are included in the class diagram.
- ➔ Business rules not included in the class diagram are described by words or using OCL (Object Constraint Language) included as UML notes.

A4. Data Modeling

- ➔ To obtain a conceptual model, **iteratively** go through these steps:
  - ➔ 1. Identify entity types (a collection of people, places, things, events, or concepts);
  - ➔ 2. Identify relationships (entities have relationships with other entities);
  - ➔ 3. Identify attributes (each entity type will have one or more data attributes);
  - ➔ 4. Apply naming conventions (team standards and guidelines applicable to data modeling).

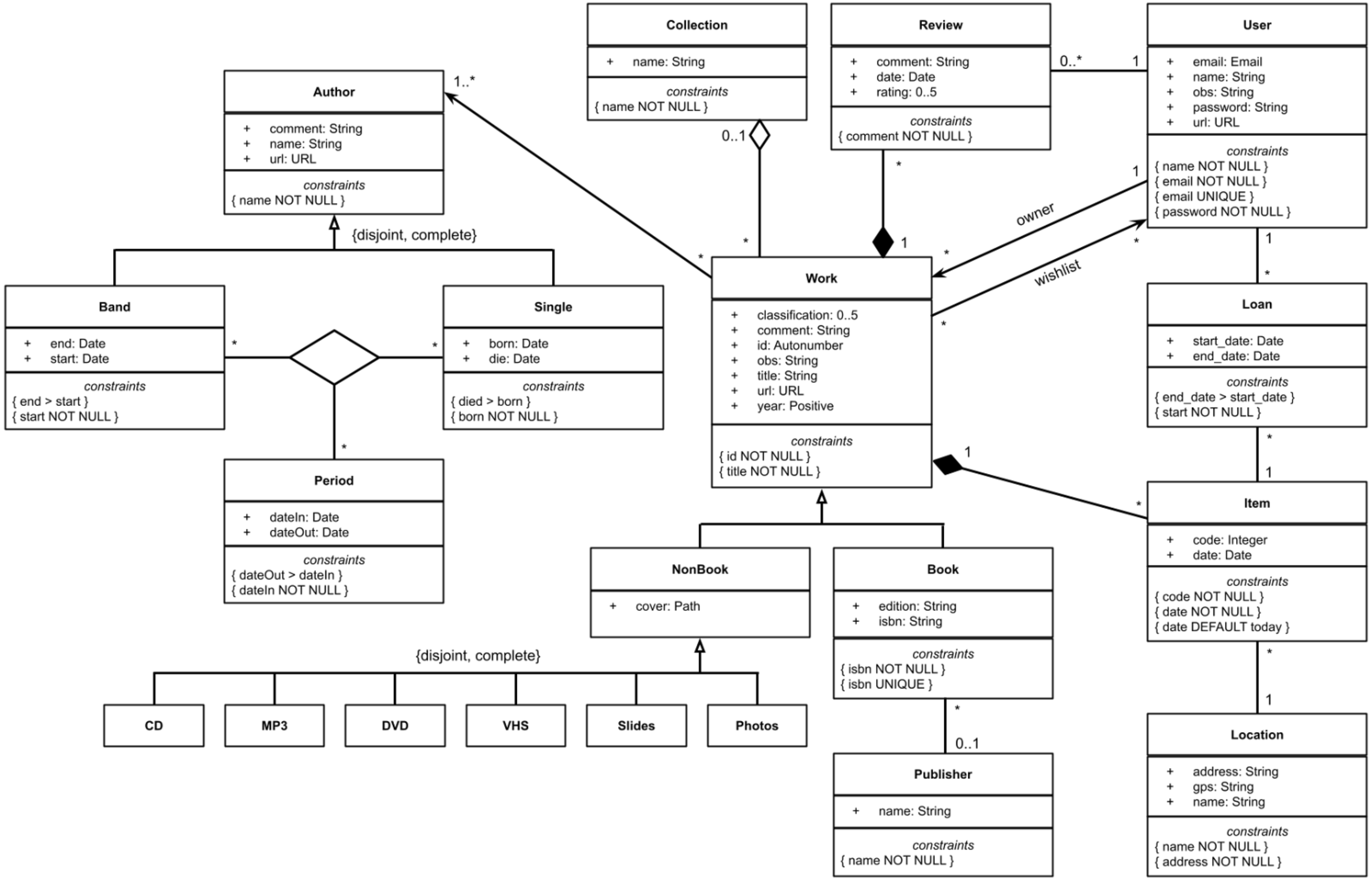
- ➔ *Scott Ambler. The Object Primer. Cambridge University Press, 3rd Edition, 2004. Section. 8.5*

A4. Data Modeling

- Data models are used for a variety of purposes, from conceptual models to physical design models
- Logical data models are used to **explore domain concepts and their relationships**
- With data modeling, you **identify data entities and assign data attributes** to them
  - *whereas with class modeling you identify classes and assign responsibilities to them*
- Then, you identify the **associations** between entities
  - relationships, inheritance, and composition
  - similar to the associations between classes

➔ *Scott Ambler. The Object Primer. Cambridge University Press, 3rd Edition, 2004. Section. 8.5*

A4. MediaLibrary Class Diagram



A4. MediaLibrary Business Rules

- ➔ BR01. A user cannot loan its own items.
- ➔ BR02. An item can only be lended by its owner.
- ➔ BR03. An item can only be loaned to one user at a time.

A4. Checklist

A4. Conceptual Data Model		
Artefact	**1.1**	The artefact reference and name are clear
	1.2	The goal of the artefact is briefly presented (1, 2 sentences)
UML	**2.1**	UML notation is consistently used
	2.2	Diagram layout is clear (visual organization)
	2.3	Classes are correctly represented
	2.4	Generalizations are correctly represented
	2.5	Associations are correctly represented
	2.6	Restrictions and business rules are correctly represented
Classes	**3.1**	Classes are presented with areas for name and attributes
	3.2	Classes do not have methods associated
	3.3	The classes support all high priority user stories defined in A2
	3.4	Class names are consistent (e.g. always singular, always in English)
Associations, multiplicity, roles	**4.1**	Automatic primary keys are not presented
	4.2	Natural keys are not used as primary keys (e.g. NIF)
	4.3	Multiplicity is defined for all associations
	4.4	Roles are used to explain how an object participates in the relationship
	4.5	Mandatory associations (not null) are indicated in the multiplicity
	4.6	In 1-1 associations, directionality is defined
	4.7	The associations support all high priority user stories defined in A2
	4.8	There is an "Authorship" association
Attributes, domains and restrictions	**5.1**	All attributes have a generic type (text, number, date, boolean)
	5.2	Attribute visibility is omitted (e.g. '+' prefix not included)
	5.3	Domains are defined for attributes that have predefined fixed values
	5.4	Not null attributes are indicated in the restrictions
	5.5	Unique attributes are indicated in the restrictions
	5.6	Restrictions related to numerical attributes are indicated (e.g. > 0)
	5.7	Restrictions related to date types are indicated (e.g. > today)
	5.8	Attributes with default values are indicated
	5.9	All generalizations have constraints defined
	5.10	All restrictions and business rules defined in A2 are included

A5. Relational Schema, Validation and Schema Refinement

- The A5 artifact contains the Relational Schema obtained by mapping from the Conceptual Data Model.
- The Relational Schema includes each relation schema, attributes, domains, primary keys, foreign keys and other integrity rules: UNIQUE, DEFAULT, NOT NULL, CHECK.
- Relation schemas are specified in the compact notation.
- In addition to this representation, the relational schema is also presented in SQL as an annex.
- To validate the Relational Schema obtained from the Conceptual Model, all functional dependencies are identified and the normalization of all relation schemas is accomplished.
- Should it be necessary, in case the scheme is not in the Boyce–Codd Normal Form (BCNF), the relational schema is refined using normalization.

A5. Relational Schema Compact Notation

- Relation schemas are specified in the compact notation:
- table1(id, attribute NN)
  table2(id, attribute → Table1 NN)
  table3(id1, id2 → Table2, attribute UK NN)
  table4((id1, id2) → Table3, id3, attribute)
- Primary keys are underlined. UK means UNIQUE and NN means NOT NULL.
- The specification of additional domains can also be made in a compact form, using the notation:
- Today DATE DEFAULT CURRENT_DATE
  Priority ENUM ('High', 'Medium', 'Low')
- **In PostgreSQL use lower case and the "snake_case" convention.**

A5. Relational Schema Mapping

Summary of Mapping Rules from Logical UML Models to Relational Schemas

Translated from:

UML – Metodologias e Ferramentas CASE, Vol. 1, 2ª Edição, pp. 314-315

Alberto Silva e Carlos Videira, Centro Atlântico (2005)

Rule 1	Classes are mapped into relation schemas
Rule 2	Class attributes are mapped to attributes of relations.
Rule 3	Operations of classes are generally not mapped. They can nevertheless be mapped to *stored procedures,* stored and executed in the global context of the database involved.
Rule 4	Objects are mapped into tuples of one or more relations.
Rule 5	Each object is uniquely identified. If the identification of an object is defined **explicitly** by the OID *(object identifier*) stereotype, associated with one or more attributes, this attribute is mapped to primary key in the relation schema. Otherwise, we assume **implicitly** that the corresponding primary key is derived from a new attribute with the name of the relation and common suffix (e.g. "PK", "ID").
Rule 6:	The mapping of many-to-many associations involves the creation of a new relation schema, with attributes acting together as primary key, and individually as foreign key for each of the schemas derived from the classes involved.
Rule 7:	The mapping of one-to-many associations involves the introduction, in the relation schema corresponding to the class that has the constraint "many", of a foreign key attribute for the other schema.

Alberto Manuel Rodrigues da Silva e Carlos Alberto Escaleira Videira, UML, metodologias e ferramentas CASE, 2ª Edição, Volume 1, Centro Atlântico Editora, Maio 2005.

A5. Relational Schema Mapping

Rule 8:	The mapping of one-to-one associations has in general two solutions. The first corresponds to the fusion of the attributes of the classes involved in one common schema. The second solution is to map each of the classes in the corresponding schema and choose one of the schemas as the most suitable for the introduction of a foreign key attribute for the other schema. This attribute should also be defined as unique within that schema.
Rule 9:	Association navigability in general has no impact on the mapping process. The exception lies in one-to-one associations, when they are complemented with navigation cues it helps in the selection of the schema that should include the foreign key attribute.
Rule 10:	Aggregation and composition associations have a minimal impact on the mapping process, which may correspond to the definition of constraints cascade ("CASCADE") in changing operations and/or removal of tuples.
Rule 11:	The mapping of generalization associations in general presents three solutions. The first solution consists in crushing the hierarchy of classes in a single schema corresponding to the original superclass. This solution is appropriate when there is a significant distinction in the structure of sub-classes and/or when the semantics of their identification is not strong. The second solution is to consider only schemas corresponding to the sub-classes and duplicate the attributes of the super-class in these schemas; in particular it works if the super-class is defined as abstract. The third solution is to consider all the schemas corresponding to all classes of the hierarchy, resulting in a mesh of connected schemas and maintained at the expense of referential integrity rules. This solution has the advantage of avoiding duplication of information among different schemas, but suggests a dispersion of information by various schemas, and might involve a performance penalty in query operations or updating of data by requiring the execution of various join operations (i.e. "JOIN") and/or validation of referential integrity.

Alberto Manuel Rodrigues da Silva e Carlos Alberto Escaleira Videira, UML, metodologias e ferramentas CASE, 2ª Edição, Volume 1, Centro Atlântico Editora, Maio 2005.

A5. MediaLibrary Relational Schema

R01	user(<u>id</u>, email **UK NN**, name **NN**, obs, password **NN**, img, is_admin **NN**)
R02	author(<u>id</u>, name **NN**, img)
R03	collection(<u>id</u>, name **NN**)
R04	work(<u>id</u>, title **NN**, obs, img, year **NN CK** year > 0, id_user -> user **NN**, id_collection -> collection)
R05	author_work(<u>id_author</u> -> author, <u>id_work</u> -> work)
R06	nonbook(<u>id_work</u> -> work, type **NN CK** type **IN** Types)
R07	publisher(<u>id</u>, name **NN**)
R08	book(<u>id_work</u> -> work, edition, isbn **UK NN**, id_publisher -> publisher)
R09	location(<u>id</u>, name **NN**, address **NN**, gps)
R10	item(<u>id</u>, id_work -> work **NN**, id_location -> location **NN**, code **NN**, date **NN DF** Today)
R11	loan(<u>id</u>, id_item -> item **NN**, id_user -> user **NN**, start_date **NN**, end_date **NN CK** end > start)
R12	review(<u>id_user</u> -> user, <u>id_work</u> -> work, date **NN DF** Today, comment **NN**, rating **NN CK** rating > 0 AND rating < = 5)
R13	wish_list(<u>id_user</u> -> user, <u>id_work</u> -> work)

A5. Schema Validation and Refinement

- ➔ To validate the Relational Schema obtained from the Conceptual Model,
  - ➔ **all functional dependencies** are identified and
  - ➔ the normalization of all relation schemas is accomplished.
- ➔ Should it be necessary, in case the scheme is not in the Boyce–Codd Normal Form (BCNF), the relational schema is refined using **normalization**.

A5. Problems of Redundancy

- **Redundancy** is at the root of several problems associated with relational schemas:
  - redundant storage;
  - insert / delete / update anomalies.
- Integrity constraints, in particular **functional dependencies**, can be used to identify schemas with such problems and to suggest refinements.
- Main refinement technique: decomposition (replacing ABCD with, say, AB and BCD, or ACD and ABD).

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

A5. Normal Forms

- ➔ If a relation is in a certain normal form (BCNF, 3NF, etc), it is known that certain kinds of problems are avoided / minimized.
- ➔ Boyce-Codd normal form (BCNF) is a slightly stronger version of the third normal form (3NF). It deals with certain types of anomalies not addressed by the 3NF.
- ➔ A relation R is in BCNF if and only if, for every functional dependency, at least one of the following conditions hold:
  - ➔ $X \rightarrow Y$ is a trivial functional dependency ($Y \subseteq X$),
  - ➔ X is a superkey for schema R.

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

A5. MediaLibrary Validation and Schema Refinement

Table R01 (user)	
Keys: {id}, {email}	
Functional Dependencies	
FD0101	{id} → {email, name, obs password, img, is_admin}
FD0102	{email} → {id, name, obs, password, img, is_admin}
Normal Form	BCNF

Table R02 (author)	
Keys: {id}	
Functional Dependencies	
FD0201	{id} → {name, img}
Normal Form	BCNF

Table R05 (author_work)	
Keys: {id_author, id_work}	
Functional Dependencies	*none*
Normal Form	BCNF

A5. MediaLibrary SQL Code

➔ The A5 artefact only includes types and tables creation statements in SQL.

➔ The SQL creation script is expanded in the A6 to include indexes, triggers, and transactions.

➔ Test the SQL creation script in the production server

➔ Include it as a file in the repository, with a reference in the wiki.

A5. Checklist

A5. Relational Schema		
Artefact	1.1	The artefact reference and name are clear
	1.2	The goal of the artefact is briefly presented (1, 2 sentences)
Schema	2.1	The compact notation is correctly used
	2.2	Each relation has a unique reference
	2.3	Relation names are lowercase and in snake_case when needed
	2.4	All UML classes are mapped
	2.5	All class attributes are mapped
	2.6	All associations are mapped
	2.7	All relations have a PK
	2.8	No natural keys are used as PK
	2.9	All FK attributes reference a relation
	2.10	In 1-1 associations, a FK is used considering the directionality
	2.11	In 1-N associations, a FK is used
	2.12	In N-N associations, a relation is defined with a composite PK of two FKs
	2.13	Generalisations are correctly mapped and the choices well justified
	2.14	Domains are defined and used if necessary
Integrity rules	3.1	All NN attribute restrictions are included
	3.2	All UK attribute restrictions are included
	3.3	All date attributes have adequate restrictions
	3.4	All numeric attributes have adequate restrictions

Schema validation	4.1	Schema validation section is included
	4.2	For each relation, all candidate keys are listed
	4.3	For each relation, all FD are listed
	4.4	Each relation's normal form is identified
	4.5	The schema's normal form is identified and a justification is provided
SQL Code	5.1	The SQL script is included
	5.2	The SQL script contains the creation statements
	5.3	The SQL script cleans up the current database state
	5.4	The SQL script is cleaned (e.g. excluded from export comments)
	5.5	Code highlighting is used for readability
	5.6	All domains are included in the SQL script
	5.7	All relations are included in the SQL script
	5.8	PK are defined as SERIAL
	5.9	FK are not defined as SERIAL
	5.10	The SQL script works without errors
	5.11	SQL script is included in the group's repository
	5.12	The production database (at db.fe.up.pt) has been set up with the SQL script

A6. Indexes, Triggers, Transactions and Database Population

- ➔ This artefact contains the physical schema of the database,
  - ➔ the identification and characterization of the indexes,
  - ➔ the support of data integrity rules with triggers,
  - ➔ the definition of the database user-defined functions,
  - ➔ and the identification and characterization of the database transactions.
- ➔ This artefact also includes the complete database creation script, including all SQL code necessary to define all integrity constraints, indexes, triggers and transactions.
- ➔ Also, the database creation script and the database population script should be included as separate elements.

PostgreSQL

PostgreSQL

- ➔ PostgreSQL is the database management system used in LBAW.
- ➔ PostgreSQL is a free and open-source RBDMS that follows a client-server paradigm.
- ➔ A PostgreSQL production environment is available, which must be used in the application's production version, at db.fe.up.pt
  - ➔ Each group has a user account lbaw21gg, and a database lbaw21gg.
- ➔ This service is managed by UPdigital and is only available using the VPN to FEUP.

Setup PostgreSQL Connection

- To configure a connection to the database, use the following settings:
  - Host: db.fe.up.pt
    Port: 5432
    Database: lbaw21gg
    Username: lbaw21gg
    Password: <group password, given in class>
- For development, groups can use a local PostgreSQL instance through Docker containers.
- Instructions can be found at: https://git.fe.up.pt/lbaw/template-postgresql

PostgreSQL Clients

- PostgreSQL clients exist for all major operating system environments.
  - https://wiki.postgresql.org/wiki/PostgreSQL_Clients
- pgAdmin, used in LBAW, is one of the most popular clients - www.pgadmin.org
  - Binary packages exist, but simply use Docker to quickly setup a local instance.
- A command line interface client is available with the psql command.
  - Connect with: `psql -h db.fe.up.pt -d lbaw21gg -U lbaw21gg`

About the PostgreSQL Production Environment (**important!**)

- A PostgreSQL database contains one or more schemas, which in turn contains one or more tables.
- All databases contain a public schema, which is used as default.
- In PostgreSQL's command line interface, you can view the current active schema with: `show search_path;`
- To change the schema for the current session use: `SET search_path TO <schema>;`
- **In the PostgreSQL setup at FEUP (db.fe.up.pt), the public schema is shared between all accounts**,
  - Tables created in the public schema are visible to all users (although not accessible).
    If you look at the tables in the publish schema, you will find a long list of tables.
  - **It is important to not use the public schema and instead create a schema with the name of your group (lbaw21gg)**.
- To create this schema, use the following command: `CREATE SCHEMA <lbaw21gg>;`
- To always use this schema as the default in your project, add the following line to the beginning of your SQL scripts.
  - `SET search_path TO <lbaw21gg>;`

References

Bibliography and Further Reading

- ➔ Scott Ambler, The Object Primer, Cambridge University Press, 3rd Edition, 2004.
- ➔ Alberto Rodrigues da Silva, Carlos Videira, UML — Metodologias e Ferramentas CASE, 2ª Edição, Centro Atlântico Editora, Maio 2005.
- ➔ Raghu Ramakrishnan, Johannes Gehrke. Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003.

Lab Class #3

- ➔ Start the EBD component.
- ➔ Develop and discuss the conceptual data model (A4).
- ➔ Develop and discuss the relational schema (A5).

- ➔ Connect to the group's database at db.fe.up.pt.
- ➔ Setup a local development environment for PostgreSQL.