Database Assignment

By Wilhelmina Ndapewa Onyothi Nekoto
And Laura Wölbeling

Task 1) Description of Use Case "Sewing Pattern Library"

The Database System is designed to manage a collection of sewing patterns. Each pattern is identified by a unique id. It has a name (which is not unique), an author and a level of difficulty. Furthermore, each pattern consists of a number of pieces and has a source (e.g. a certain book or magazine).

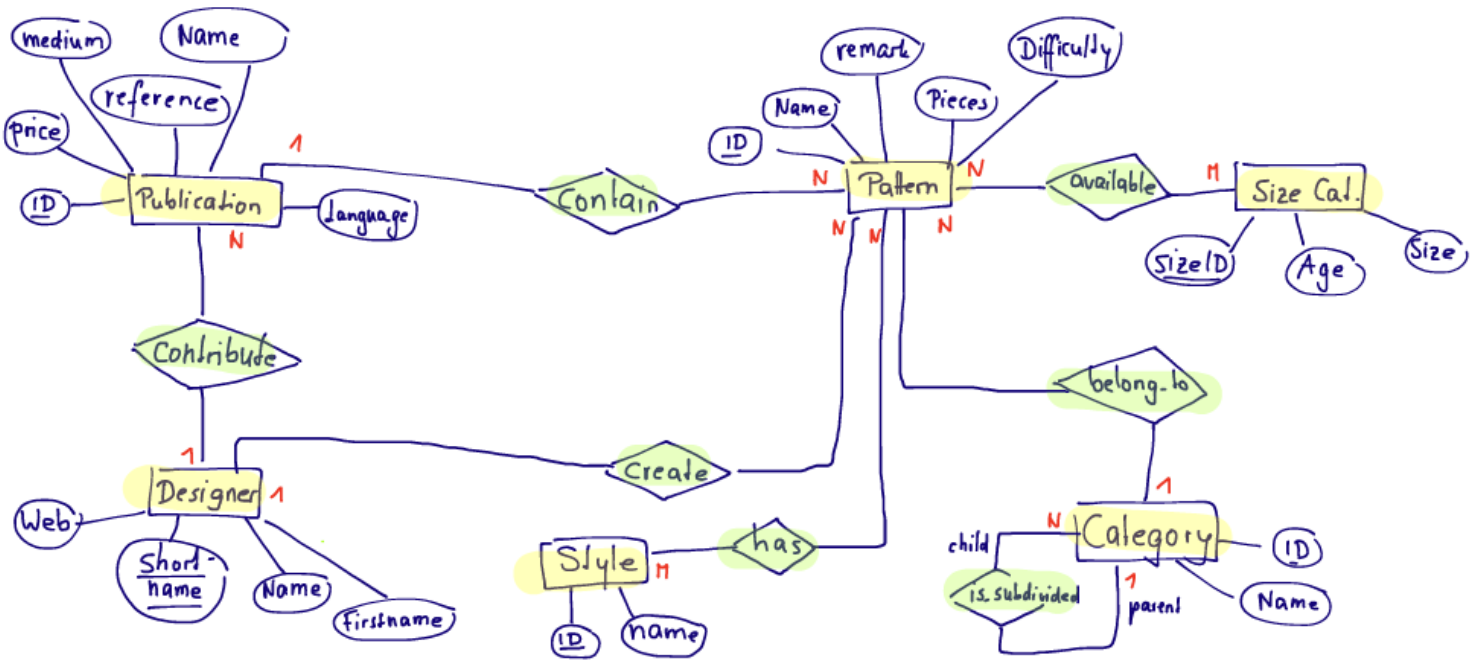
The patterns are categorized: each belongs to one category. Categories are structured hierarchical, so one category can be the parent of another category. Each category can have only one parent, but multiple children. Categories are identified by a number and have a name.

Most patterns are available for many sizes (either children's or adults'), but some are only single-size patterns or don't have a size (like accessories). The available sizes and the information on whether it refers to a child or a grown person should be stored.

Patterns can be included in magazines or books, or they can be in a digital file (ebook). There can only be one source for a pattern. We want to store information about the type/format, the source's name and author and its language, as well as a reference number, which could be either a book ISBN or a magazine issue number or something else.

The creator of a pattern is the designer. We want to store information about that person, such as her first name family name, a web address and a short name, which is the person's unique identifier.

Task 2) ER diagram



Task 3) Implementation design

a) Entity types to tables

Pattern: {[patternID: int, name: text, pieces: int, difficulty: int, remark: text]}

Style: {[styleID: int, name: text, description: text]}

Size: {[sizeID: int, agegroup: text, size: text]}

Category: {[catID: int, name: text]}

Designer: {[shortname: text, name: text, firstname: text, web: text]}

Publication: {[publD: int, name: text, language: text, medium: text, reference: text, price: int]}

b) Relation types -> tables

Contribute: {[designer: text, publD: int]}

Create: {[designer: text, patternID: int]}

Contain: {[patternID: int, publD: int]}

HasStyle: {[patternID: int, styleID: int]}

AvailableSize: {[patternID: int, sizeID: int]}

forFabric: {[patternID: int, fabric: text]}

belongCat: {[catID: int, patternID: int]}

isSubdivided: {[parent: int, child: int]}

c) Improvements / tables consolidation

Pattern: {[patternID: int, name: text, pieces: int, difficulty: int, remark:text, publD: int, catID: int]}

Style: {[styleID: int, name: text, description: text]}

Sizes: {[sizeID: int, agegroup: text, size: text]}

Category: {[catID: int, name: text, parent: int]}

Designer: {[shortname: text, name: text, firstname: text, web: text]}

Publication: {[publD: int, name: text, language: text, medium: text, reference: text, price: int, designer: txt]}

HasStyle: {[patternID: int, styleID: int]}

HasSize: {[patternID: int, sizeID: int]}

~~belongCat: {[catID: int, patternID: int]}~~

~~isSubdivided: {[parent: int, child: int]}~~

~~Contribute: {[designerID: int, publD: int]}~~

~~Create: {[designerID: int, patternID: int]}~~

~~Contain: {[patternID: int, publD: int]}~~

Task 4) Scripts

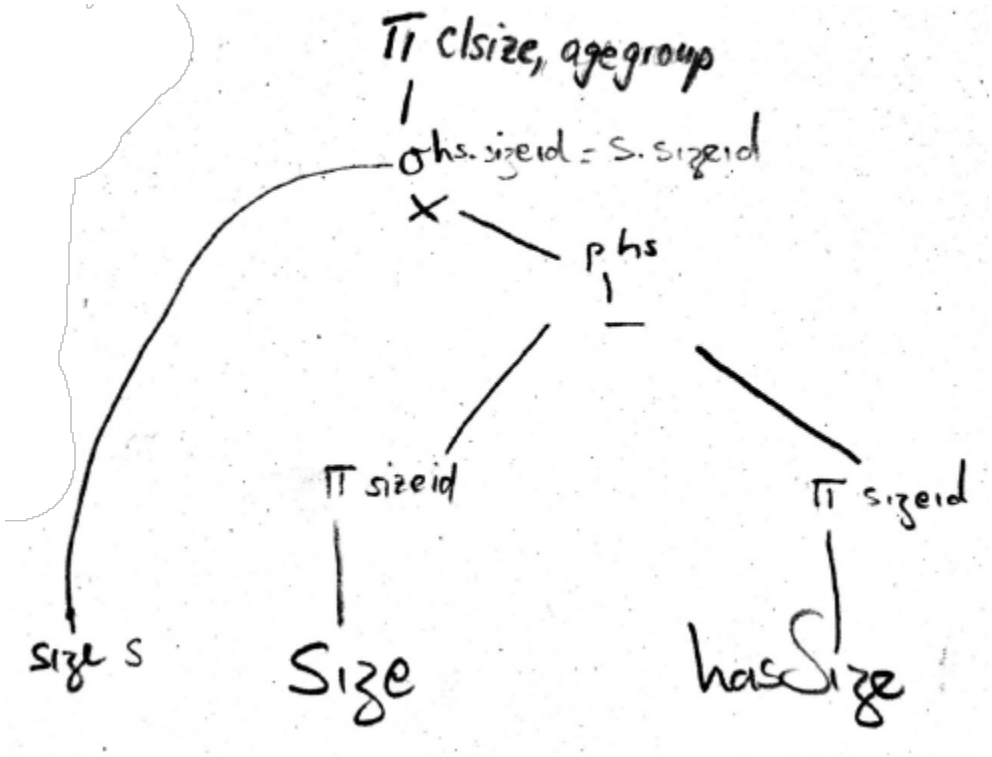
See droptables.sql, types.sql, tables.sql and data.sql

Task 5) SQL Queries

/* a) 1x operator tree and query with EXCEPT/ UNION/ INTERSECT */

/* show sizes, for which we do not have a pattern */

```
SELECT clsize, agegroup FROM sizes WHERE sizeid IN
        ((SELECT sizeid FROM sizes)
        EXCEPT
        (SELECT sizeid FROM hasSize));
```



/* b) 2x cartesian product or Join */

/* show all categories with subcategories */

```
SELECT p.name AS category, c.parentid AS catnr, c.name AS subcategory, c.catid AS subcatnr
FROM category c, category p WHERE c.parentid = p.catid;
```

/* show all publication with the patterns they contain */

```
SELECT publication.name as publication, pattern.name AS pattern FROM pattern, publication
WHERE pattern.pubid = publication.pubid;
```

/* c) 2x with set operators IN, NOT IN, ALL or EXISTS*/

/* select all foreign-language patterns */

```
SELECT name AS pattern, pieces, difficulty FROM pattern WHERE pubid NOT IN (SELECT pubid
FROM publication WHERE language = 'de');

/* select designer who has the pattern with the smallest amount of pieces */

SELECT firstname, designer.name as lastname, shortname

FROM publication, designer, pattern

WHERE publication.designer = designer.shortname

        AND publication.pubid = pattern.pubid

        AND    pieces <= ALL (SELECT pieces FROM pattern);


/* d) 2x with GROUP BY and aggregation function */

/* list all publications with the number of patterns included */

SELECT publication.name as publication, count(pattern.name) AS patterns FROM pattern,
publication WHERE pattern.pubid = publication.pubid GROUP BY publication.name;

/* What is the average number of pieces? */

SELECT avg(pieces)AS average_pieces FROM pattern;


/* e) 1x with HAVING-clause */

/* select all designers, who created 2 or more patterns */

SELECT shortname, count(pattern.patternid) FROM publication, designer, pattern

WHERE publication.designer = designer.shortname

        AND publication.pubid = pattern.pubid

GROUP BY shortname

HAVING count(pattern.patternid) >= 2;
```

Task 6) command-line interface

Written in JAVA -> see db_program.zip