CoLInS 2021

"An Approach and Software Prototype for Translation of Natural Language Business Rules into Database Structure"

Andrii Kopp

Ph.D. in Information Technology,
Senior Lecturer

kopp93@gmail.com

Dmytro Orlovskyi

Ph.D. in Information Technology,
Associate Professor

orlovskyi.dm@gmail.com

Sergey Orekhov

Ph.D. in Information Technology,
Associate Professor

sergey.v.orekhov@gmail.com

Department of Software Engineering and Management Information Technology

Faculty of Computer Science and Software Engineering

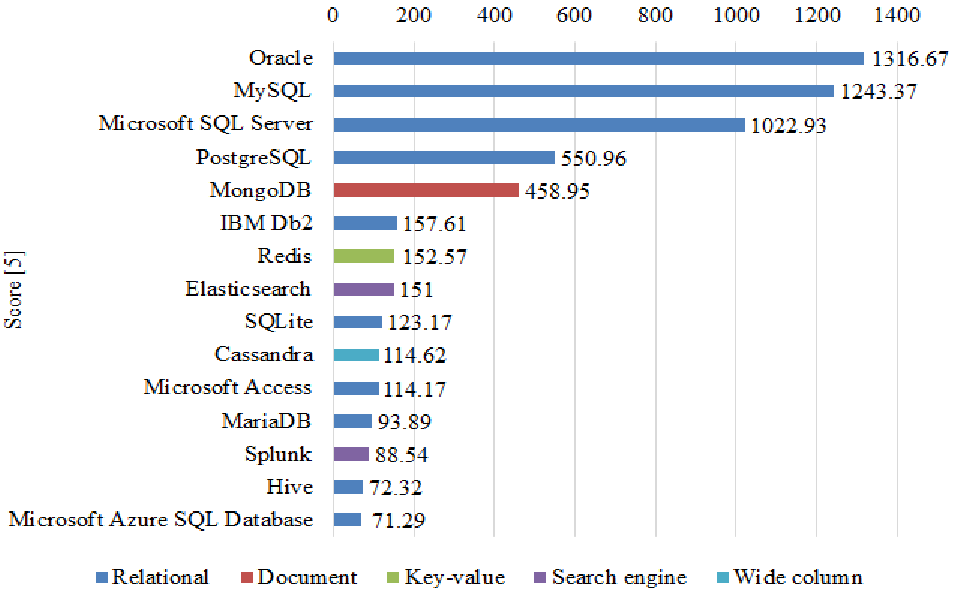
Motivation

- Database design is the part of almost any software engineering project and it may require special engineers who have advanced database construction skills.
- Database design, implementation, and support in agile projects is done by the same team members who are usually involved in server-side programming.
- Lack of special training and time to consider the database design more carefully leads to potential design flaws or even mistakes in a database schema.
- Occurrence of such problems may be prevented by specialized tools that support database design activities when translating gathered requirements into database objects.
- Hence, we propose an approach and a prototype of software tool to translate database design requirements into scripts for database schema generation and its further tuning by responsible project members.

Popularity of Relational Databases

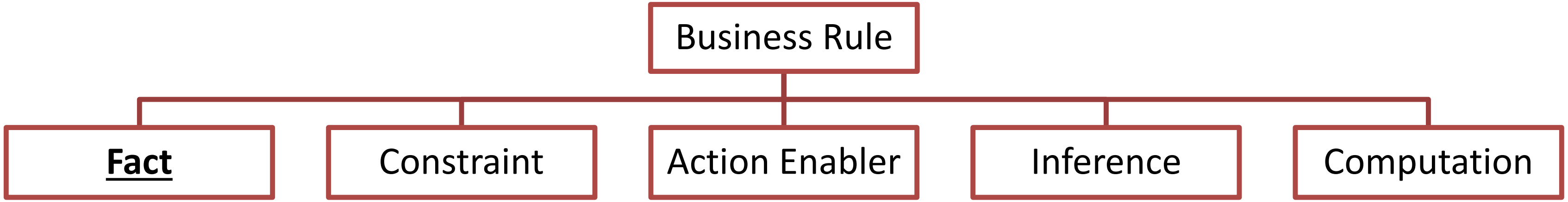
According to the DB-Engines Ranking, the five most popular DBMS are:

- Oracle (relational DBMS, also supports document and graph models).
- MySQL (relational DBMS, also supports document model).
- Microsoft SQL Server (relational DBMS, also supports document and graph models).
- PostgreSQL (relational DBMS, also supports document model).
- MongoDB (document model).



Business Rules in Database Design

- Business rules are used as sources for correct discovery of entities, attributes, constraints, and relationships.
- Business rules are brief, precise, and unambiguous textual descriptions of policies, processes, and principles within a certain organization.
- The main sources of business rules are people and documentation within organization: managers of different levels, company policies, or process manuals.



- Facts are statements that define entities, attributes, and relationships within data models.

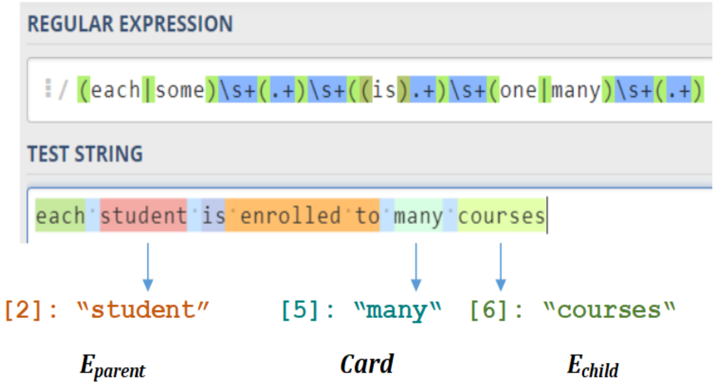
Business Rules Extraction

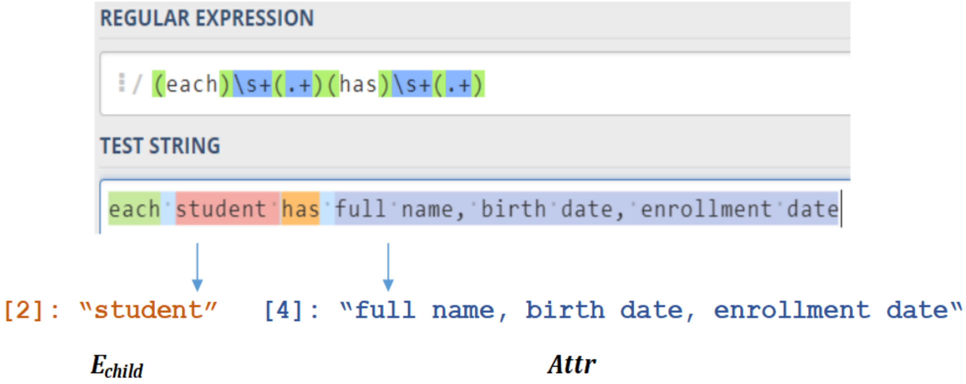
Business rules that describe:

- relationships
- entities and their attributes

$$< relationship\_business\_rule >::= \{each|some\} \ < parent\_entity > \{is \ < relation >\} \{one|many\} \ < child\_entity >.$$

$$< entity\_business\_rule >::= \{each\} \ < entity > \{has\} \{< attribute >, \ldots\}.$$



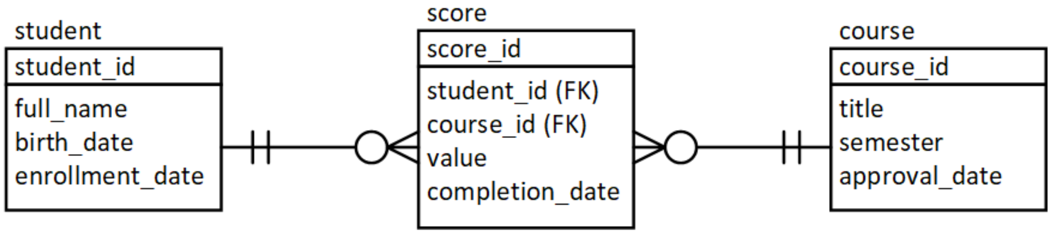


Database Scripts Generation

Business Rules ⇒ Relational Model ⇒ SQL Scripts

Each student has full name, student card id, birth date, enrollment date. Each student is given by many scores. Each course has title, semester, approval date. Each course is evaluated by many scores. Each score has value, completion date.

student → ⟨*PK* = {student_id}, *FK* = ∅, *Cols* = {full_name, student_card_id, birth_date, enrollment_date}⟩
score → ⟨*PK* = {score_id}, *FK* = {student_id, course_id}, *Cols* = {value, completion_date}⟩
course → ⟨*PK* = {course_id}, *FK* = ∅, *Cols* = {title, semester, approval_date}⟩




```
CREATE TABLE `student` ( `student_id` INTEGER,
`full_name` VARCHAR(255), `student_card_id`
VARCHAR(255), `birth_date` DATETIME,
`enrollment_date` DATETIME);
CREATE TABLE `score` ( `score_id` INTEGER,
`student_id` INTEGER, `course_id` INTEGER, `value`
DECIMAL(8,2), `completion_date` DATETIME);
CREATE TABLE `course` ( `course_id` INTEGER, `title`
VARCHAR(255), `semester` DECIMAL(8,2),
`approval_date` DATETIME);
```

```
ALTER TABLE `student` MODIFY `student_id`
INTEGER AUTO_INCREMENT PRIMARY KEY;
ALTER TABLE `score` MODIFY `score_id` INTEGER
AUTO_INCREMENT PRIMARY KEY;
ALTER TABLE `course` MODIFY `course_id` INTEGER
AUTO_INCREMENT PRIMARY KEY;
ALTER TABLE `score` MODIFY `student_id` INTEGER
NOT NULL;
ALTER TABLE `score` ADD FOREIGN KEY
(`student_id`) REFERENCES `student`(`student_id`);
ALTER TABLE `score` MODIFY `course_id` INTEGER
NOT NULL;
ALTER TABLE `score` ADD FOREIGN KEY (`course_id`)
REFERENCES `course`(`course_id`);
```

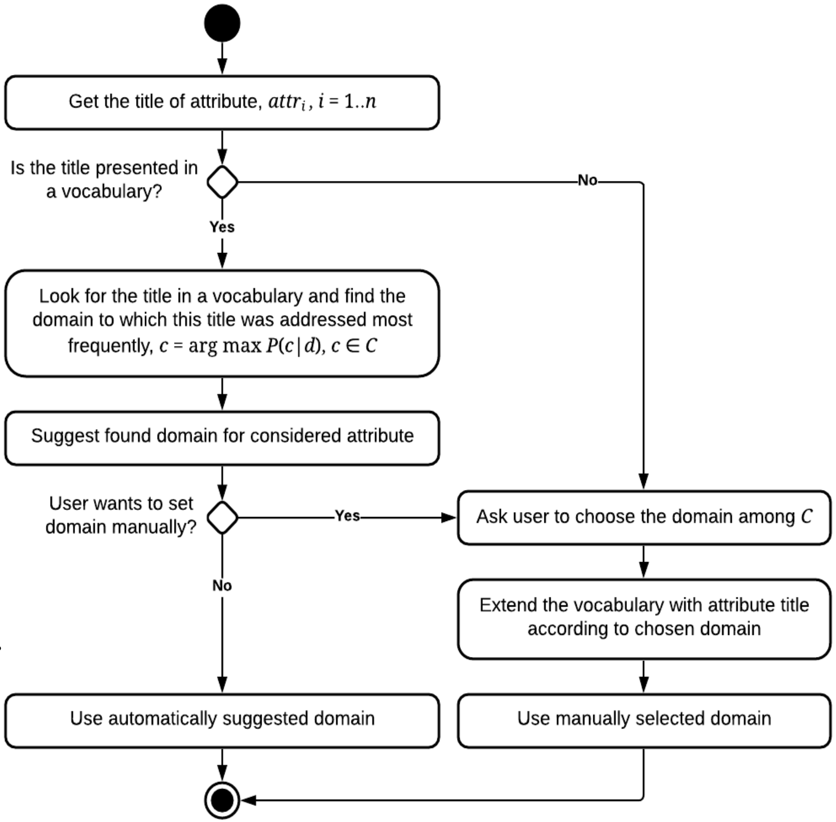
Column Domains Suggestion

Naïve Bayes approach could be used to suggest domains, since it is fast enough for real-time multiclass predictions and does not need large volumes of training data:

$$c = \arg\max_{c \in C} P(c|d),$$

- C is the set of domains used as classes, $C = \{DateTime, Number, Text, Boolean\}$;
- $P(c|d)$ is the number of titles, which belong to the domain $c \in C$, matched to the attribute d.

DateTime ⇒ DATETIME, Number ⇒ DECIMAL, Text ⇒ VARCHAR, Boolean ⇒ TINYINT(1)



Alternate Keys Suggestion

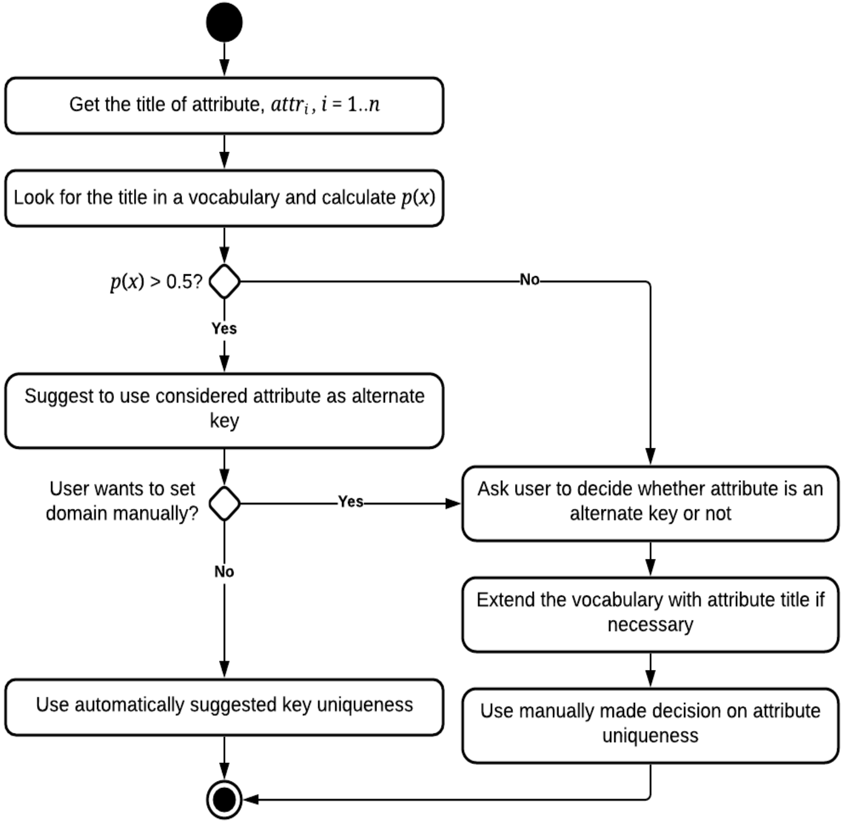
Logistic activation function could be used to formalize suggestion of UNIQUE indexes:

$$p(x) = \frac{1}{1 + e^{-x}},$$

- x is the frequency of attribute title occurrences among the vocabulary of alternate key titles.

If a value obtained using the logistic model is greater than 0.5, then such column might be used as unique index.

student → ⟨PK = {student_id}, FK = ∅, $Cols$ = {full_name, student_card_id, birth_date, enrollment_date}⟩ ⇒ ALTER TABLE `student` ADD UNIQUE (`student_card_id`);

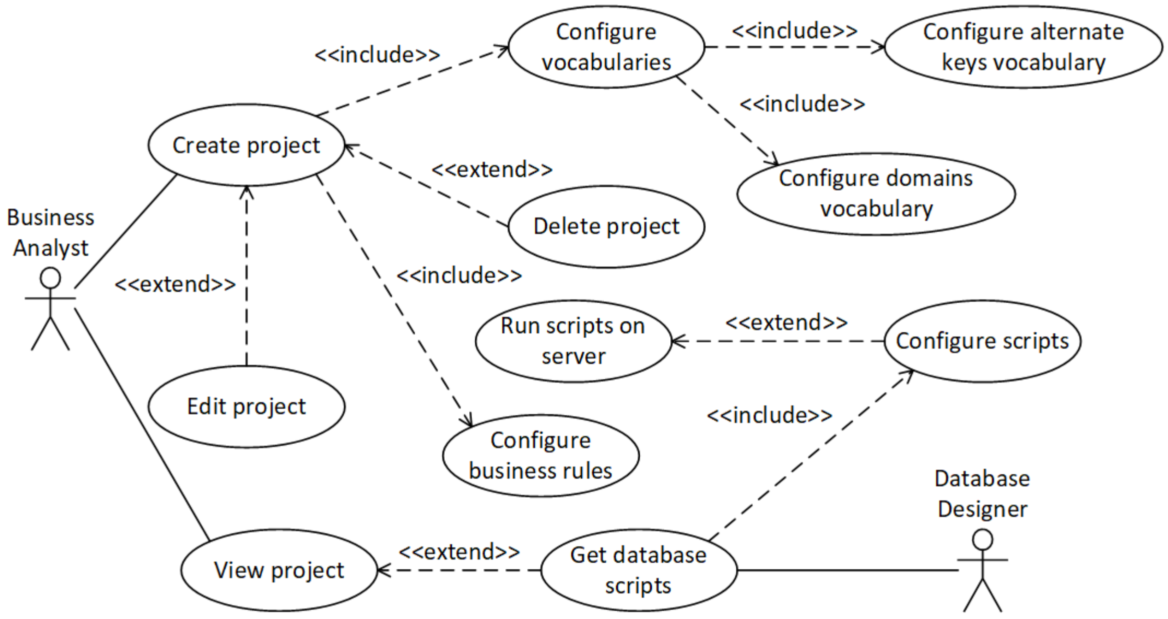


Software Tool Prototype: Use Cases

The software tool is supposed to be used by business analyst and database designer roles.

Main features:

- Work with projects
- Work with business rules within projects
- Work with vocabularies for column domains and alternate keys suggestion
- SQL scripts generation for each of projects



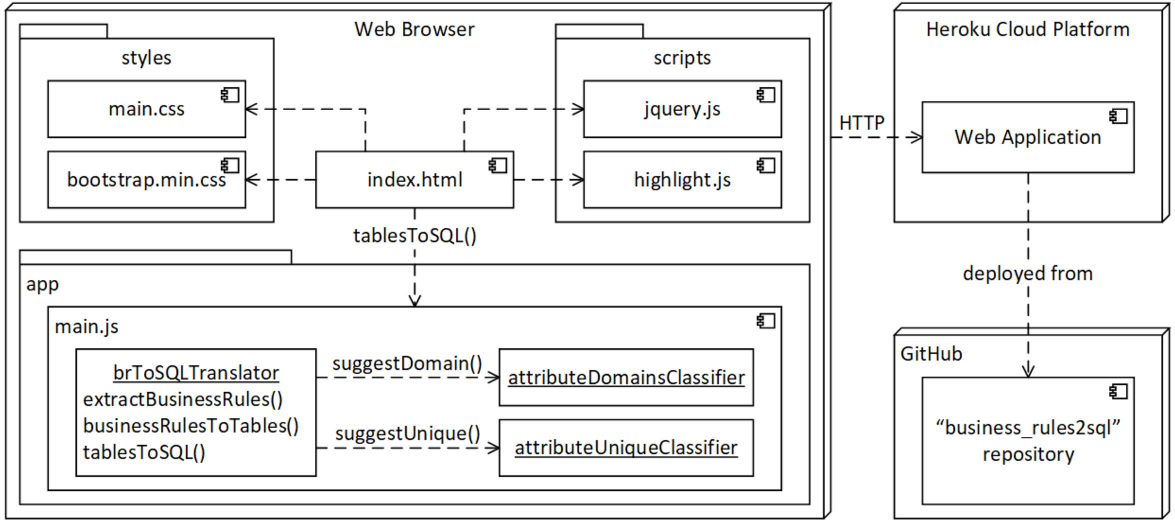
Software Tool Prototype: Architecture

Software prototype consists of:

- Home web page (index.html)
- Styles (main.css)
- Scripts (main.js)

Software prototype depends on:

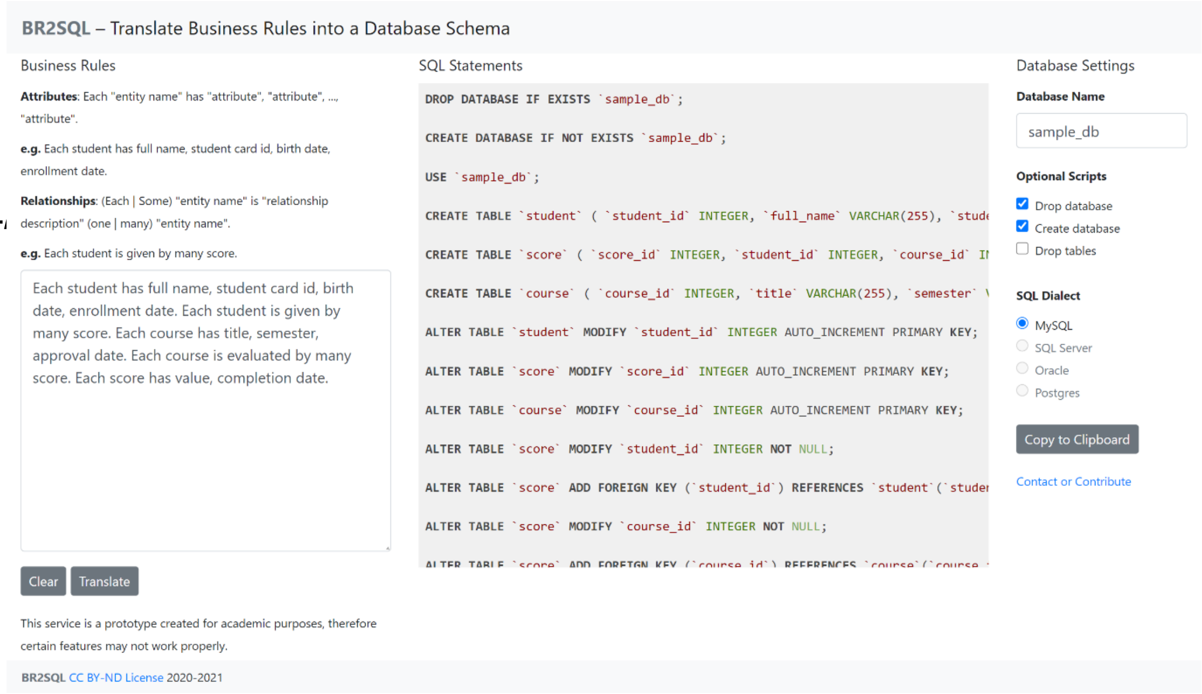
- Bootstrap library for user interface (bootstrap.min.css)
- jQuery library for DOM (Document Object Model) operations
- Source code highlighting library (highlight.js)



Software Tool Prototype: Homepage Example

User interface elements are:

- Text area for business rules
- Control buttons (clear text area, translate rules into SQL, copy code to clipboard)
- Generated code area
- Text input for database name
- Check boxes for optional settings
- Radio button to select SQL dialect



Example of Business Rules Translation into Database Structure

Sample set of business rules:

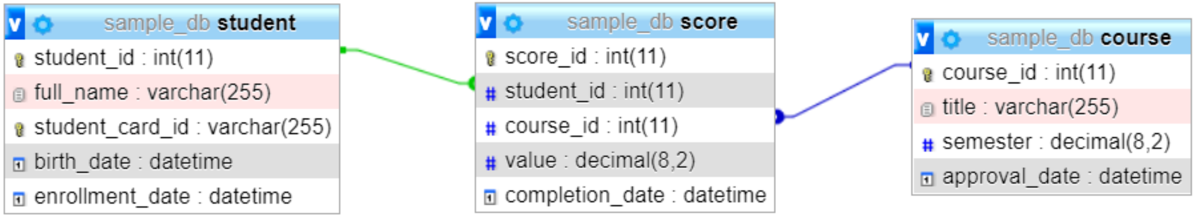
- “Each student has full name, student card id, birth date, enrollment date”.
- “Each student is given by many scores”.
- “Each course has title, semester, approval date”.
- “Each course is evaluated by many scores. Each score has value, completion date”.

```sql
DROP DATABASE IF EXISTS `sample_db`;
CREATE DATABASE IF NOT EXISTS `sample_db`;
USE `sample_db`;

CREATE TABLE `student` ( `student_id` INTEGER, `full_name` VARCHAR(255), `student_card_id`
VARCHAR(255), `birth_date` DATETIME, `enrollment_date` DATETIME);
CREATE TABLE `score` ( `score_id` INTEGER, `student_id` INTEGER, `course_id` INTEGER, `value`
DECIMAL(8,2), `completion_date` DATETIME);
CREATE TABLE `course` ( `course_id` INTEGER, `title` VARCHAR(255), `semester` DECIMAL(8,2),
`approval_date` DATETIME);

ALTER TABLE `student` MODIFY `student_id` INTEGER AUTO_INCREMENT PRIMARY KEY;
ALTER TABLE `score` MODIFY `score_id` INTEGER AUTO_INCREMENT PRIMARY KEY;
ALTER TABLE `course` MODIFY `course_id` INTEGER AUTO_INCREMENT PRIMARY KEY;
ALTER TABLE `score` MODIFY `student_id` INTEGER NOT NULL;
ALTER TABLE `score` ADD FOREIGN KEY (`student_id`) REFERENCES `student`(`student_id`);
ALTER TABLE `score` MODIFY `course_id` INTEGER NOT NULL;
ALTER TABLE `score` ADD FOREIGN KEY (`course_id`) REFERENCES `course`(`course_id`);

ALTER TABLE `student` ADD UNIQUE (`student_card_id`);
```



Classifiers Validation

Vocabularies content after the business rules of previous example were translated

Vocabulary	Classification	Data
Column domains	DateTime	birth_date, enrollment_date, completion_date, approval_date
	Number	value, semester
	Text	full_name, student_card_id, title
Alternate keys	UNIQUE	student_card_id

Another attempt of different business rules translation has resulted into the automatically suggested domains, corresponding MySQL data types, and UNIQE alternate keys

Each person has full name, birth date.

```
CREATE TABLE `person` ( `person_id` INTEGER, `full_name`
VARCHAR(255), `birth_date` DATETIME);
ALTER TABLE `person` MODIFY `person_id` INTEGER AUTO_INCREMENT
PRIMARY KEY;
```

Each progress paper has student card id, title, semester, value, completion date.

```
CREATE TABLE `progress_paper` ( `progress_paper_id` INTEGER,
`student_card_id` VARCHAR(255), `title` VARCHAR(255), `semester`
DECIMAL(8,2), `value` DECIMAL(8,2), `completion_date` DATETIME);
ALTER TABLE `progress_paper` MODIFY `progress_paper_id` INTEGER
AUTO_INCREMENT PRIMARY KEY;
ALTER TABLE `progress_paper` ADD UNIQUE (`student_card_id`);
```

Each payment has value, approval date, completion date.

```
CREATE TABLE `payment` ( `payment_id` INTEGER, `value`
DECIMAL(8,2), `approval_date` DATETIME, `completion_date`
DATETIME);
ALTER TABLE `payment` MODIFY `payment_id` INTEGER
AUTO_INCREMENT PRIMARY KEY;
```

Validation of Database Operability, Integrity, and Consistency

SELECT * FROM `student`

student_id	full_name	student_card_id	birth_date	enrollment_date
1	Patrick G. Harrington	239-09	1988-07-28 00:00:00	2003-07-30 00:00:00
2	Stephen J. Dantzler	389-13	1993-04-30 00:00:00	2011-07-28 00:00:00
3	David B. Bailey	421-94	1998-02-10 00:00:00	2017-07-30 00:00:00

SELECT * FROM `score`

score_id	student_id	course_id	value	completion_date
1	3	1	84.00	2018-01-20 00:00:00
2	3	2	91.00	2018-07-03 00:00:00
3	3	3	78.00	2019-01-12 00:00:00
4	3	4	74.00	2019-06-21 00:00:00
5	3	5	81.00	2020-01-11 00:00:00

SELECT * FROM `course`

course_id	title	semester	approval_date
1	Introduction to Software Engineering	1	2017-05-30 00:00:00
2	Basics of Object-Oriented Programming	2	2017-11-30 00:00:00
3	Introduction to Databases	3	2018-05-30 00:00:00
4	Design of Databases	4	2018-11-30 00:00:00
5	Architecture and Design of Software	5	2019-05-30 00:00:00
6	Design of Information Systems	6	2019-11-30 00:00:00
7	Software Quality	7	2020-05-30 00:00:00
8	Systems Analysis	8	2020-05-30 00:00:00

INSERT INTO score (student_id, course_id, value, completion_date) VALUES (4, 1, 94, '2020-01-16');

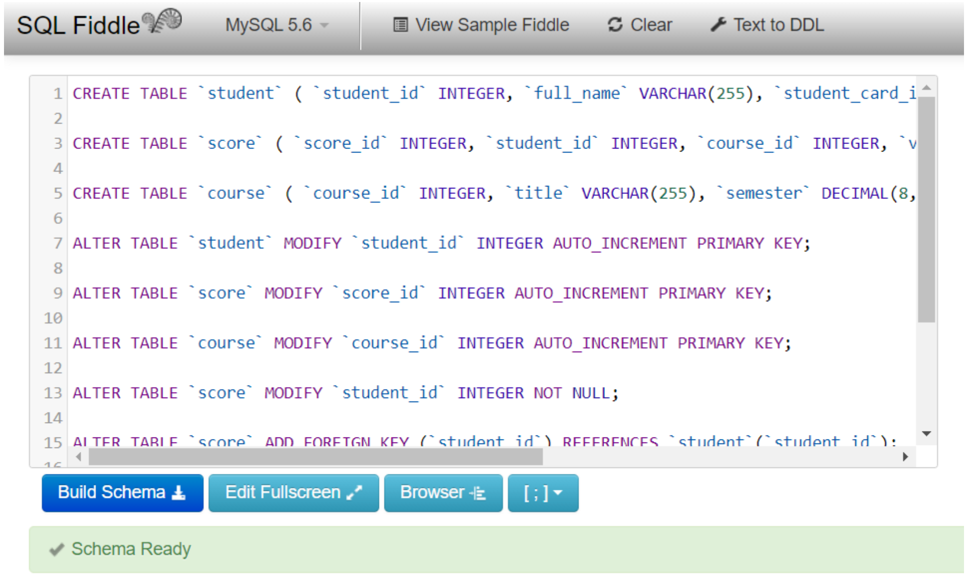
```
#1452 - Cannot add or update a child row: a foreign key constraint fails (`sample_db`.`score`, CONSTRAINT `score_ibfk_1` FOREIGN KEY (`student_id`) REFERENCES `student` (`student_id`))
```

INSERT INTO student (full_name, student_card_id, birth_date, enrollment_date) VALUES ('Larry S. Ruiz', '389-13', '1999-01-02', '2018-07-27');

```
#1062 - Duplicate entry '389-13' for key 'student_card_id'
```

SQL Statements Verification and Evaluation of Database Structure

Generated SQL code was verified by MySQL when executed, which is also proven by SQL Fiddle



Database structure was evaluated using the Data Model Scorecard metrics, among which we have chosen the most interesting by our opinion:

Metric	Total score	Model score	Value
How complete is the model?	15	3	0.20
How structurally sound is the model?	15	10	0.67
How well does the metadata match the data?	10	8	0.80

Given scores reflect incompleteness of business rules, presence non-atomic attribute, and inaccurate data types (VARCHAR and DECIMAL)

Conclusion and Future Work

- We have presented the approach and software tool prototype for translation of natural language business rules into database structure
- The approach is based on textual descriptions written in a special way, so they could be processed and relational model that contains entities, attributes, and relationships could be obtained
- Obtained relational mapping then translated into DDL scripts used to create database schema in a relational DBMS
- Presented results demonstrate working software prototype and its usage to translate sample set of business rules into MySQL database schema
- Future work may include usage of more advanced machine learning and natural language processing methods in order to suggest attribute domains and alternate keys
- Also in future it is planned to consider support of business logic requirements and referential integrity constraints

THANK YOU FOR ATTENTION!