CS 340
Final Project
Tatsiana Clifton

Online Survey Database

Outline

The database for the project is the online surveys database. Online surveys are popular in different areas. Many companies use surveys to get a feedback from users about their products and services. On the other hand, online surveys have many benefits such as lower cost and higher speed of gathering data. On the other hand, it requires developing an effective storage for data. Obviously, a database is an excellent solution for that.

Most online surveys may require storing various data such as questions, available answers, actual data, and information about respondents. Data for online surveys that is stored in a database should be available for easy filtering depending on the purpose for using it. Thus, it is essential to create a database that will allow working efficiently with it.

Database Outline in Words

The online survey database includes the following entries:

- survey;
- question;
- respondent;
- answer.

Each survey uniquely identified by id, it has a name, a description and the opening (creation) time. The name of each survey is unique.

Every question uniquely identified by id, it has some content. Each question's content is unique.

Every respondent uniquely identified by id, he/she has an email address, the gender, the country of living, the date of birth. There is no the same email addresses. In other words, every email address is distinct.

Each answer uniquely identified by id and it has some answer content.

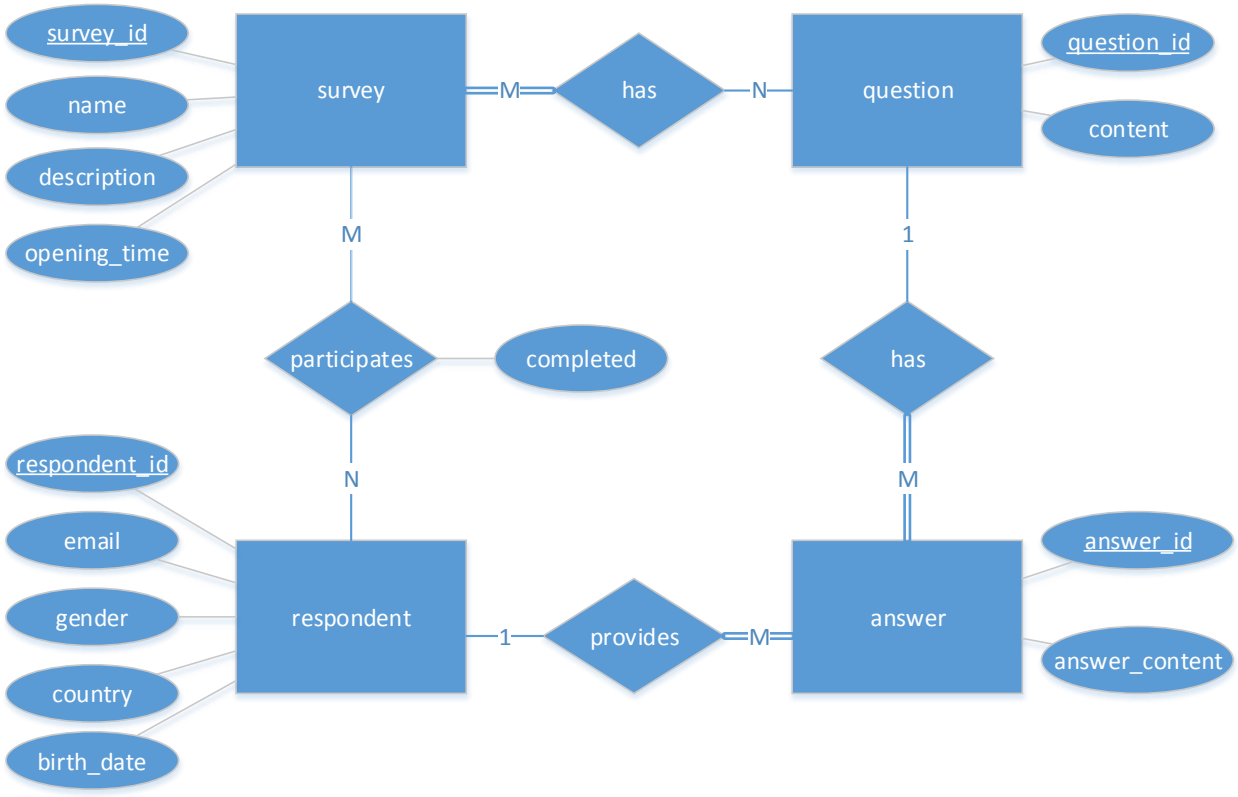
Surveys have questions. Each survey has at least one question. One survey can have many questions. One question can be used in many surveys. However, some questions can be unused.

Respondents complete surveys. One respondent can complete many surveys. Many respondents can complete one survey. There can be respondents who did not complete any surveys yet. There can be some surveys that were not completed by any respondents yet.

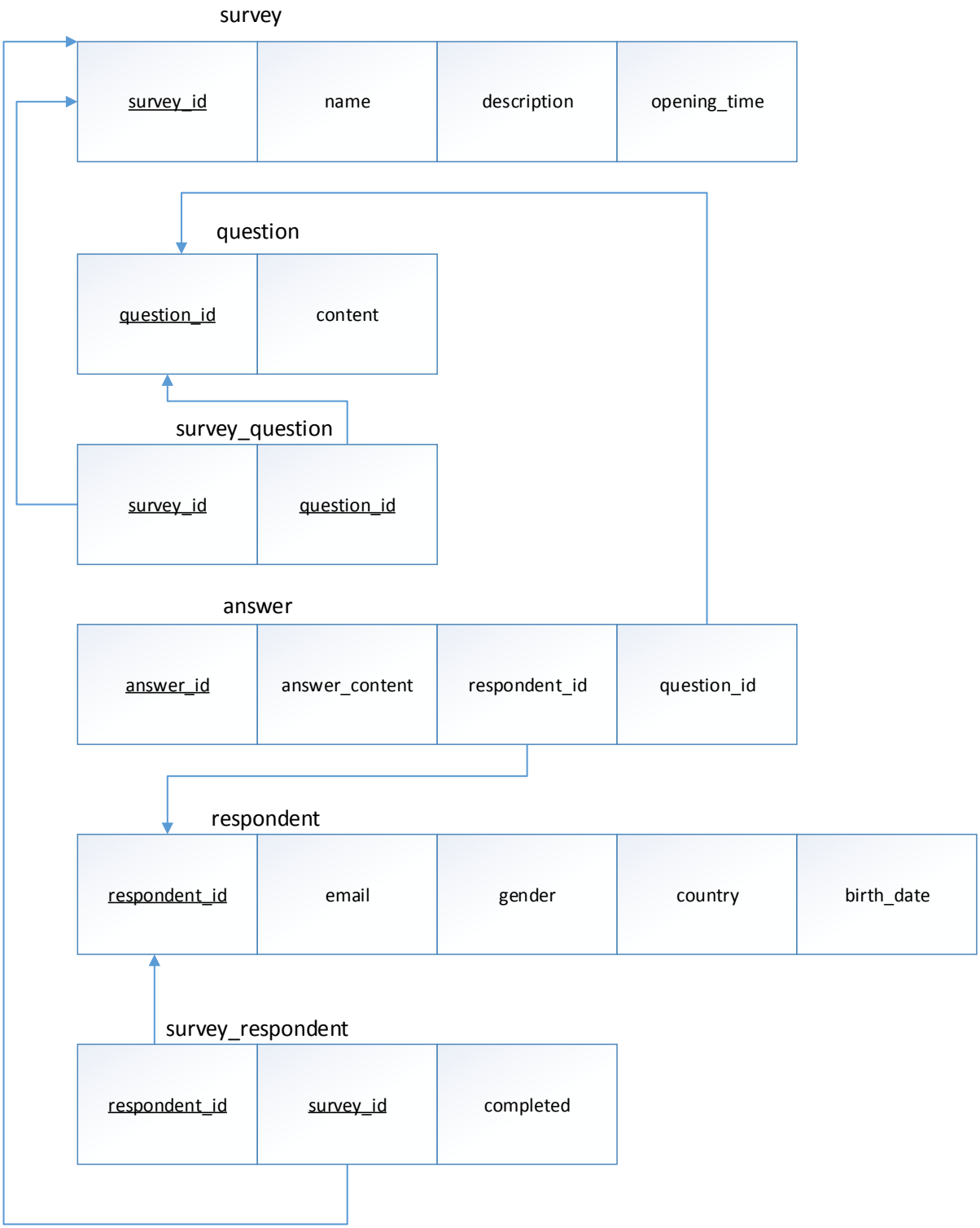
Respondents provide answers. One user can have many answers. Every answer belongs to a certain respondent. Each answer must have a respondent. There are some respondents who did not provide any answers yet.

Each answer is for exactly one certain question. Each question can have many answers or do not have any answers.

ER Diagram of Database



Database Schema



Tables Creation Queries and General Use Queries

Tables Creation

```
CREATE TABLE `survey`(
    `survey_id` int NOT NULL AUTO_INCREMENT,
    `name` varchar(255) NOT NULL,
    `description` varchar(255) NOT NULL,
    `openning_time` date,
    PRIMARY KEY (`survey_id`),
    UNIQUE (`name`)
) ENGINE=InnoDB;

CREATE TABLE `question`(
    `question_id` int NOT NULL AUTO_INCREMENT,
    `content` varchar(500) NOT NULL,
    PRIMARY KEY (`question_id`),
    UNIQUE (`content`)
) ENGINE=InnoDB;

CREATE TABLE `survey_question`(
    `survey_id` int,
    `question_id` int,
    PRIMARY KEY (`survey_id`, `question_id`),
    FOREIGN KEY (`survey_id`) REFERENCES `survey`(`survey_id`) ON
DELETE CASCADE,
    FOREIGN KEY (`question_id`) REFERENCES `question`(`question_id`)
ON DELETE CASCADE
) ENGINE=InnoDB;

CREATE TABLE `respondent`(
    `respondent_id` int NOT NULL AUTO_INCREMENT,
    `email` varchar(255) NOT NULL,
    `gender` varchar(1),
    `country` varchar(255),
    `birth_date` date,
    PRIMARY KEY (`respondent_id`),
    UNIQUE (`email`)
) ENGINE=InnoDB;

CREATE TABLE `survey_respondent`(
    `survey_id` int,
    `respondent_id` int,
    `completed` date NOT NULL,
    FOREIGN KEY (`survey_id`) REFERENCES `survey`(`survey_id`) ON
DELETE CASCADE,
    FOREIGN KEY (`respondent_id`) REFERENCES
`respondent`(`respondent_id`) ON DELETE CASCADE
) ENGINE=InnoDB;
```

```
CREATE TABLE `answer`(
    `answer_id` int NOT NULL AUTO_INCREMENT,
    `question_id` int,
    `respondent_id` int,
    `answer_content` int NOT NULL,
    PRIMARY KEY (`answer_id`),
    FOREIGN KEY (`respondent_id`) REFERENCES
`respondent`(`respondent_id`) ON DELETE CASCADE,
    FOREIGN KEY (`question_id`) REFERENCES `question`(`question_id`)
ON DELETE CASCADE
) ENGINE=InnoDB;
```

General Use

```
INSERT INTO `survey` (name, description, opening_time)
VALUES ([name], [description], [time]);

INSERT INTO `question` (content)
VALUES ([content]);

INSERT INTO `survey_question` (survey_id, question_id)
VALUES ((SELECT survey_id FROM `survey` WHERE name = [name]), (SELECT
question_id FROM `question` WHERE content = [content]));

INSERT INTO `respondent` (email, gender, country, birth_date)
VALUES ([email], [gender], [country], [birth]);

INSERT INTO `survey_respondent` (survey_id, respondent_id, completed)
VALUES ((SELECT survey_id FROM `survey` WHERE name = [name]), (SELECT
respondent_id FROM `respondent` WHERE email = [email]), [completed]);

INSERT INTO `answer` (respondent_id, question_id, answer_content)
VALUES ((SELECT respondent_id FROM `respondent` WHERE email =
[email]), (SELECT question_id FROM `question` WHERE content =
[content]), [answer_content]);

SELECT q.content AS question_text FROM question q
INNER JOIN survey_question sq ON sq.question_id = q.question_id
INNER JOIN survey s ON s.survey_id = sq.survey_id
WHERE s.survey_id = [id]
ORDER BY q.content DESC;

SELECT s.name AS survey_name, s.description AS survey_description,
COUNT(temp.age) AS number_of_respondents FROM survey s
LEFT JOIN
    (SELECT r.respondent_id, TIMESTAMPDIFF(YEAR, birth_date,
CURDATE()) AS age, sr.survey_id FROM respondent r
    INNER JOIN survey_respondent sr ON sr.respondent_id =
r.respondent_id) AS temp
ON temp.survey_id = s.survey_id
WHERE temp.age >= [age1] AND temp.age <= [age2];
```

```
SELECT s.name AS name, s.description AS description,
COUNT(sr.respondent_id) AS respondents FROM survey s
RIGHT JOIN survey_respondent sr ON sr.survey_id = s.survey_id
GROUP BY s.name;

SELECT temp.name AS survey_name, temp.description AS
survey_description, temp.respondents AS number_of_respondents
FROM
    (SELECT s.name AS name, s.description AS description,
COUNT(sr.respondent_id) AS respondents FROM survey s
     RIGHT JOIN survey_respondent sr ON sr.survey_id = s.survey_id
     GROUP BY s.name) AS temp
WHERE temp.respondents = (SELECT MAX(temp2.respondents)
    FROM
    (SELECT s.name AS name, s.description AS description,
COUNT(sr.respondent_id) AS respondents FROM survey s
    RIGHT JOIN survey_respondent sr ON sr.survey_id = s.survey_id
    GROUP BY s.name) AS temp2);

DELETE FROM survey
WHERE survey_id = [id];

UPDATE question SET content = [newContent]
WHERE question_id = [id];
```