A9: Main accesses to the database and transactions

1. Main Accesses

Below are the main accesses to the database in **Presto**, separated by modules, with each access associated to the resources it is used at.

M01: Authentication and Static Pages

SQL101	Check if a member or admin with given credentials exists
Web Resource	R102, R105

```
-- If count > 0, user with given credentials exists
SELECT count(*)
FROM (SELECT id
FROM member
WHERE email = $email AND password = $password
UNION ALL
SELECT id
FROM admin
WHERE email = $email AND password = $password) AS all_platform_users;
```

SQL102	Register a new member
Web Resource	R105

```
INSERT INTO member (username, email, password, name) VALUES ($username, $email,
$password, $name);
```

M02: Members

SQL201	Obtain a member’s profile information
Web Resource	R201

```
SELECT member.name, bio, profile_picture, score, is_moderator, country.name
FROM member INNER JOIN country ON member.country_id = country.id
WHERE username = $username;
```

SQL202	Obtain a member’s most recent questions
Web Resource	R201

```
SELECT title, content, date, string_agg(topic.name, ', ')
FROM question INNER JOIN member ON question.author_id = member.id INNER JOIN
question_topic ON question.id = question_id INNER JOIN topic ON topic.id = topic_id
WHERE member.id = 10
GROUP BY title, content, date
ORDER BY date DESC
LIMIT 10;
```

SQL203	Obtain a member's most recent answers
Web Resource	R201

```
SELECT title, answer.content, answer.date, profile_picture, member.name,
string_agg(topic.name, ', ')
FROM answer INNER JOIN question ON answer.question_id = question.id INNER JOIN member ON
answer.author_id = member.id INNER JOIN question_topic ON question.id =
question_topic.question_id INNER JOIN topic ON topic.id = topic_id
WHERE member.id = 10
GROUP BY title, answer.content, answer.date, profile_picture, member.name
ORDER BY date DESC
LIMIT 10;
```

SQL204	Update a member's profile information
Web Resource	R204

```
UPDATE member
SET username = $new_username, name = $name, password = $password, bio = $bio,
profile_picture = $profile_picture
WHERE username = $old_username;
```

SQL205	Obtain a member's followers
Web Resource	R205

```
SELECT m1.name, m1.profile_picture, m1.score, m1.bio, m1.is_moderator
FROM follow_member INNER JOIN member AS m1 ON follow_member.follower_id = m1.id INNER
JOIN member m2 ON follow_member.following_id = m2.id
WHERE m2.username = $username;
```

SQL206	Obtain a member's notifications
Web Resource	R215

```
SELECT type, content, date, read
FROM notification INNER JOIN member ON notification.member_id = member.id
WHERE username = $username
ORDER BY date DESC;
```

SQL207	Mark a notification as read

Web Resource	R216

```
UPDATE notification
SET read = true
WHERE id = $id;
```

M03: Content (Questions, Answers and Comments)

SQL301	Obtain a question’s details
Web Resource	R302

```
SELECT member.name, member.profile_picture, question.title, question.content,
question.date, (SELECT count(*) FROM question_rating WHERE question_id = $id AND rate =
1) as positive_votes, (SELECT count(*) FROM question_rating WHERE question_id = $id AND
rate = -1) as negative_votes
FROM question INNER JOIN member ON question.author_id = member.id
WHERE question.id = $id;
```

SQL302	Add a new question
Web Resource	R303

```
INSERT INTO question (title, content, date, author_id) VALUES ($title, $content, $date,
$author_id);
```

SQL303	Associate a topic to a question
Web Resource	R303

```
INSERT INTO question_topic (question_id, topic_id) VALUES ($question_id, $topic_id);
```

SQL304	Open/Close (mark as solved) question
Web Resource	R306

```
UPDATE question
SET solved = $solved
WHERE id = $id;
```

SQL305	Rate a question
Web Resource	R307

```
INSERT INTO question_rating (member_id, question_id, rate) VALUES ($member_id,
$question_id, $rate);
```

SQL306	Obtain an answer’s details
Web Resource	R310

```
SELECT member.name, member.profile_picture, answer.content, answer.date, (SELECT
count(*) FROM answer_rating WHERE answer_id = $id AND rate = 1) as positive_votes,
(SELECT count(*) FROM answer_rating WHERE answer_id = $id AND rate = -1) as
negative_votes
FROM answer INNER JOIN member ON answer.author_id = member.id
WHERE answer.id = $id;
```

SQL307	Add a new answer
Web Resource	R311

```
INSERT INTO answer (content, date, author_id, question_id) VALUES ($content, $date,
$author_id, $question_id);
```

SQL308	Rate an answer
Web Resource	R314

```
INSERT INTO answer_rating (member_id, answer_id, rate) VALUES ($member_id, $answer_id,
$rate);
```

SQL309	Obtain a comment’s details
Web Resource	R316

```
SELECT member.name, member.profile_picture, comment.content, comment.date, (SELECT
count(*) FROM comment_rating WHERE comment_id = $id AND rate = 1) as positive_votes,
(SELECT count(*) FROM comment_rating WHERE comment_id = $id AND rate = -1) as
negative_votes
FROM comment INNER JOIN member ON comment.author_id = member.id
WHERE comment.id = $id;
```

SQL310	Add a new comment
Web Resource	R317

```
INSERT INTO comment (content, date, author_id, question_id, answer_id) VALUES ($content,
$date,  $author_id, $question_id, $answer_id);
```

SQL311	Rate a comment
Web Resource	R320

```
INSERT INTO comment_rating (member_id, comment_id, rate) VALUES ($member_id,
$comment_id, $rate);
```

M04: Topics

SQL401	Obtain topic information
Web Resource	R402

```
SELECT name, picture, description, count(*) as numFollowers
FROM topic INNER JOIN follow_topic ON topic_id = id
WHERE name = $name
GROUP BY name, picture, description;
```

SQL402	Obtain the most recent questions associated to a topic
Web Resource	R402

```
SELECT member.name, member.profile_picture, question.title, question.content,
question.date, string_agg(topic.name, ', ')
FROM question INNER JOIN member ON question.author_id = member.id INNER JOIN
question_topic ON question.id = question_id INNER JOIN topic ON topic_id = topic.id
WHERE question.id IN (SELECT question_id FROM question_topic INNER JOIN topic ON
topic_id = topic.id WHERE topic.name = 'Outdoors')
GROUP BY member.name, member.profile_picture, question.title, question.content,
question.date
ORDER BY question.date DESC
LIMIT 10;
```

SQL403	Obtain topics related to a specific topic (i.e., topics that show up most frequently side by side with it on questions)
Web Resource	R402

```
SELECT name
FROM topic INNER JOIN question_topic ON topic.id = question_topic.topic_id
WHERE question_id IN (SELECT question_id FROM question_topic INNER JOIN topic ON
topic_id = id WHERE name = $name) AND topic.name <> $name
GROUP BY name
ORDER BY count(*) DESC
LIMIT 5;
```

SQL404	Add a new topic to the system
Web Resource	R403

```
INSERT INTO topic (name, description, picture) VALUES ($name, $description, $picture);
```

SQL405	Follow a topic

Web Resource	R408

```
INSERT INTO follow_topic (topic_id, member_id) VALUES ($topic_id, $member_id);
```

SQL406	Unfollow a topic
Web Resource	R408

```
DELETE FROM follow_topic WHERE topic_id = $topic_id AND member_id = $member_id;
```

M05: Index and Search

SQL501	Obtain trending topics
Web Resource	R502, R503

```
SELECT name
FROM topic INNER JOIN question_topic ON topic.id = question_topic.topic_id
GROUP BY name
ORDER BY count(*) DESC
LIMIT 5;
```

SQL502	Obtain top members
Web Resource	R502, R503

```
SELECT username, name, profile_picture, score, is_moderator
FROM member
ORDER BY score DESC
LIMIT 10;
```

SQL503	Obtain questions ordered by ‘top’ status (most answers)
Web Resource	R502, R503

```
SELECT member.name, member.profile_picture, question.title, question.date
FROM question INNER JOIN member ON question.author_id = member.id INNER JOIN answer on
answer.question_id = question.id
GROUP BY member.name, member.profile_picture, question.title, question.date
ORDER BY count(*) DESC;
```

SQL504	Obtain questions ordered by date (most recent first)
Web Resource	R502, R503

```
SELECT member.name, member.profile_picture, question.title, question.date
FROM question INNER JOIN member ON question.author_id = member.id
ORDER BY question.date DESC;
```

SQL505	Obtain questions that match a given input
Web Resource	R505

```
SELECT member.name, member.profile_picture, question.title, question.content,
question.date
FROM question INNER JOIN member ON question.author_id = member.id
WHERE question.search @@ plainto_tsquery(‘english’, $search_input)
ORDER BY ts_rank(question.search, plainto_tsquery(‘english’, $search_input) DESC;
```

SQL506	Obtain members whose username matches a given input
Web Resource	R505

```
SELECT username, name, profile_picture, bio
FROM member
WHERE username ILIKE '%' || $input || '%';
```

SQL507	Obtain topics whose name matches a given input
Web Resource	R505

```
SELECT name, picture, description
FROM topic
WHERE name ILIKE '%' || $input || '%';
```

SQL507	Obtain questions whose date falls in a specific range
Web Resource	R505

```
SELECT member.name, member.profile_picture, question.title, question.content,
question.date
FROM question INNER JOIN member ON question.author_id = member.id
WHERE question.date BETWEEN $beginning_date AND $end_date;
```

SQL508	Obtain answers whose date falls in a specific range
Web Resource	R505

```
SELECT member.name, member.profile_picture, answer.content, answer.date, question.title
FROM answer INNER JOIN member ON answer.author_id = member.id INNER JOIN question ON
answer.question_id = question.id
WHERE answer.date BETWEEN $beginning_date AND $end_date;
```

M06: Administration

SQL601	Obtain all members whose username matches input
Web Resource	R602

```
SELECT username, name, email
FROM member
```

```
WHERE username ILIKE '%' || $input || '%';
```

SQL602	Promote member to have moderator privileges
Web Resource	R604

```
UPDATE member SET is_moderator = TRUE WHERE username = $input;
```

SQL603	Ban member
Web Resource	R603

```
UPDATE member SET is_banned = TRUE WHERE username = $input;
```

2. Transactions

Below are the necessary transactions to ensure data integrity in **Presto**.

T01	Associate topics to question when it is created, while creating topics that don't already exist
Isolation Level	Serializable Read Write
Justification	Transaction involves checking topic table multiple times: for each topic associated to the new question, check if it already exists and, if not, create it, associating afterwards the question to the topic in the database. Multiple questions can be associated with the same topic that has not yet been created and, therefore, there is the possibility of inconsistencies due to possible INSERTs (T1 performs first loop SELECT, followed by T2 performing whole loop for the same not yet existing topic), transaction must be serializable. Read write due to the possibility of inserting new topics and surely inserting new question_topic tuples.
Web resource	R303

```
BEGIN TRANSACTION;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ WRITE;
-- For each topic
SELECT id FROM topic WHERE name ILIKE $input;
IF NOT FOUND THEN --topic not yet in the database
INSERT INTO topic (name) VALUES ($input);
END IF;
INSERT INTO question_topic (question_id, topic_id) VALUES ($question_id, $topic_id);
-- loop end
COMMIT;
```

T02	Get full question page (question, answers, comments and respective ratings)
Isolation Level	Read Uncommitted Read Only
Justification	Transaction gathers all information about a question to show in its page (answers, comments and ratings). While gathering this information some inaccuracies might occur (e.g. we might miss a message or a rating). However, this is an inconsistency that does not cause an alarming inconvenience to the user's experience and, therefore, read uncommitted will suffice. Read only since the transaction only gathers information through SELECT statements.
Web resource	R301

```sql
BEGIN TRANSACTION;
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED READ ONLY;
-- question
SELECT member.name, member.profile_picture, question.title, question.content,
question.date, (SELECT count(*) FROM question_rating WHERE question_id = $q_id AND rate
= 1) as positive_votes, (SELECT count(*) FROM question_rating WHERE question_id = $q_id
AND rate = -1) as negative_votes
FROM question INNER JOIN member ON question.author_id = member.id
WHERE question.id = $q_id;

-- question comments
SELECT member.name, member.profile_picture, comment.content, comment.date
FROM comment INNER JOIN member ON comment.author_id = member.id
WHERE question_id = $q_id;

-- answers
SELECT member.name, member.profile_picture, answer.content, answer.date
FROM answer INNER JOIN member ON answer.author_id = member.id
WHERE question_id = $q_id;

-- for each answer
SELECT member.name, member.profile_picture, comment.content, comment.date
FROM comment INNER JOIN member ON comment.author_id = member.id
WHERE answer_id = $a_id;
-- loop end
COMMIT;
```

Revision history

Changes made to the first submission:

1. Added associated web resources to each transaction.

GROUP1725, 26/04/2018

António Cunha Seco Fernandes de Almeida, up201505836@fe.up.pt
Bruno Manuel Nascimento Costa Galvinas Piedade, up201505668@fe.up.pt
Diogo Luis Rey Torres, up201506428@fe.up.pt
João Paulo Madureira Damas, up201504088@fe.up.pt