A9: MAIN ACCESSES TO THE DATABASE AND TRANSACTIONS-LEARNIT (Q&A)

1. MAIN ACCESS

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

1.1. MODULE M01: AUTHENTICATION

SQL101	Check if a member or admin with given credentials exists.
Web resource	R102 and R105

```
SELECT count (*)
FROM (
    SELECT id_user
    FROM "user"
    WHERE username = $username AND password = $password
)
AS all_platform_users;
```

1.2. MODULE M03: USERS

SQL201	Obtain member's profile information.
Web resource	R301

```
SELECT username, bioDescription, points, profilePhoto, name
FROM "user", rank
Where "user".id_rank = rank.id_rank AND "user".id_user =
$id_user;
```

SQL202	Update member's profile information.
Web resource	R304

```
UPDATE "user"
SET username = $username, password = $password, bioDescription =
$bioDescription,
profilePhoto = $profilePhoto, points = $points
WHERE id_user = $id;
```

SQL203	Obtain a member's most recent questions.
Web resource	R301

```
SELECT title, description, date
FROM question INNER JOIN "user" ON question.id_user =
"user".id_user
GROUP BY title, description, date
ORDER BY date DESC
LIMIT 10;
```

SQL204	Obtain member's followers.
Web resource	R301

```
SELECT id_user, username, profilePhoto, points, id_rank
FROM follow INNER JOIN "user" ON (follow.follower =
"user".id_user)
WHERE follow.following = $id_user;
```

SQL205	Obtain member's following.
Web resource	R301

```
SELECT id_user, username, profilePhoto, points, id_rank
FROM follow INNER JOIN "user" ON follow.following =
"user".id_user
WHERE follow.follower = $id_user;
```

SQL206	Obtain member's notification.
Web resource	R305

```
SELECT type, description, "date", view
FROM notification INNER JOIN "user" ON notification.id_user =
"user".id_user
WHERE username = $username
ORDER BY date DESC;
```

SQL207	Mark notification as read.
Web resource	R306

```
UPDATE notification
SET view = true
WHERE id_notification = $id;
```

1.3. MODULE M04: MAIN CONTENT

SQL401	Obtain question's details.
Web resource	R402

```
SELECT "user".username, "user".profilePhoto, question.title,
question.description, question."date", question.votes,
question.photo
FROM question INNER JOIN "user" ON (question.id_user =
"user".id_user)
WHERE question.id_question = $id_question;
```

SQL402	Add a new question.
Web resource	R404

```
INSERT INTO question (title, description, "date",votes, photo,
deleted, id_category, id_user,search)
VALUES ($title, $description, now(), 0, $photo, false,
$id_category, $id_user, NULL);
```

SQL403	Rate a question.
Web resource	R407

```
UPDATE question
SET votes = votes + $quantidade
WHERE id_question = $id_question;
```

SQL404	Obtain answer details.
Web resource	R409

```
SELECT "user".username, "user".profilePhoto, answer."text",
answer.photo, answer."date", answer.votes
FROM answer INNER JOIN "user" ON (answer.user_post =
"user".id_user)
WHERE answer.id_answer = $id_answer;
```

SQL405	Add a new answer.
Web resource	R410

```
INSERT INTO answer ("text", "date", votes, photo, deleted,
id_question, user_post)
VALUES ($"text", now(), 0, $photo, false, $id_question,
$user_post);
```

SQL406	Rate an answer.
Web resource	R413

```
UPDATE answer
SET votes = votes + $quantidade
WHERE id_answer = $id_answer;
```

SQL407	Add a new comment.
Web resource	417

```
INSERT INTO comment(firstAnswer,secondAnswer)
VALUES($firstAnswer, $secondAnswer);
```

1.4. MODULE M05: FEED AND SEARCH

SQL501	Obtain the most recent questions associated to a category.
Web resource	R503

```
SELECT "user".username, "user".profilePhoto, question.title,
question.description, question.date, category.name
FROM question
INNER JOIN "user" ON question.id_user = "user".id_user
INNER JOIN category ON question.id_category =
category.id_category
ORDER BY question.date DESC
LIMIT 10;
```

SQL502	Obtain questions ordered by date.
Web resource	R503

```
SELECT "user".username, "user".profilePhoto, question.title,
question.description, question.date
FROM question INNER JOIN "user" ON question.id_user =
"user".id_user
ORDER BY question.date DESC;
```

SQL503	Obtain questions ordered by most answers.
Web resource	R503

```
SELECT "user".username, "user".profilePhoto, question.title,
question.description, question.date
FROM question INNER JOIN "user" ON question.id_user =
"user".id_user
INNER JOIN answer on answer.id_question = question.id_question
GROUP BY "user".username, "user".profilePhoto, question.title,
question.description, question.date
ORDER BY count ("user".username, "user".profilePhoto,
question.title, question.description, question.date) DESC;
```

SQL504	Obtain members whose username matches a given input.
Web resource	R503 and R603

```
SELECT username, profilePhoto, bioDescription, email
FROM "user"
WHERE username ILIKE '%' || $input || '%';
```

SQL505	Obtain questions whose date falls in a specific range.
Web resource	R503

```
SELECT "user".username, "user".profilePhoto, question.title,
question.description, question.date
FROM question INNER JOIN "user" ON question.id_user =
"user".id_user
WHERE question.date BETWEEN $beginning_date AND $end_date;
```

SQL506	Obtain questions that match a given input.
Web resource	R503

```
SELECT id_question, title, description
FROM question
WHERE search @@ plainto_tsquery('english', $search) AND
question.deleted = false
ORDER BY ts_rank(search, plainto_tsquery('english',$search)) DESC
LIMIT 10;
```

1.5. MODULE M06: ADMINISTRATION

SQL601	Ban member.
Web resource	R606

```
UPDATE "user"
SET banned = true
WHERE "user".id_user = $id_user;
```

SQL602	Add category.
Web resource	R604

```
INSERT INTO category(name, icon)
VALUES($name, $icon);
```

2. TRANSACTIONS

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

T01	Insert a new role when a new user is created.
Isolation Level	repeatable read.
Justification	This transaction is necessary to maintain consistency, since every time a user is created it is necessary for it to have a role. If there is an error, for example when inserting the role, a rollback is issued. The isolation level is Repeatable Read because an insert on table user committed by a concurrent transaction would update id_user and therefore inconsistent data would be stored.
Web resource	R105

```
BEGIN TRANSACTION;
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

    -- Insert user
    INSERT INTO "user" (username, password, email, bioDescription,
birthdate, profilePhoto, points, id_rank, banned, deleted)
    VALUES ($username, $password, $email, NULL, $birthdate, NULL,
0, id_rank_default, false, false)
    RETURNING user.id_user INTO $user_id;

    -- Insert role
    INSERT INTO role(roleType, beginningDate, endDate , id_user)
    VALUES (member, now(), NULL, $user_id);

    COMMIT;
```

T02	Update and assign as best answer.
Isolation Level	Repeatable read.
Justification	This transaction is necessary to maintain consistency considering that we need to have concordance between the information on the answer that is chosen to be best answer and the one inserted in the table bestAnswer. The isolation level is Repeatable Read.
Web resource	R415

```
BEGIN TRANSACTION;
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

    --Select an answer
    SELECT "user".username, "user".profilePhoto, answer."text",
answer.photo,
    answer."date", answer.votes
    FROM answer INNER JOIN "user" ON (answer.user_post =
"user".id_user)
    WHERE answer.id_answer = $id_answer;

    --Assign as best answer
    INSERT INTO bestAnswer(id_bestAnswer, attributionDate, "text",
"date", deleted, active, votes, photo)
    VALUES ($id_answer, now(), $text, $date, false, true, $votes,
$photo);

    COMMIT;
```

T03	Dismiss Moderator (turn moderator into a common user).
Isolation Level	repeatable read.
Justification	This transaction is necessary to maintain consistency considering that if we don't end a user's role when assigning a new one, that user will have two simultaneous roles. If there is an error, for example when updating the date on the role, a rollback is issued. The isolation level is Repeatable Read because an insert on table role committed by a concurrent transaction would create two roles for the same user and therefore inconsistent data would be stored.
Web resource	R608

```
BEGIN TRANSACTION;
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

    --Update current role date
    UPDATE role
    SET endDate = now()
    WHERE id_user = $id;

   --Create new role
    INSERT INTO role(roleType, beginningDate, endDate , id_user)
    VALUES (member, now(), NULL, $id);

    COMMIT;
```

T04	Promote member to have moderator privileges.
Isolation Level	repeatable read
Justification	This transaction is necessary to maintain consistency considering that if we don't end a user's role when assigning a new one, that user will have two simultaneous roles. If there is an error, for example when updating the date on the role, a rollback is issued. The isolation level is Repeatable Read because an insert on table role committed by a concurrent transaction would create two roles for the same user and therefore inconsistent data would be stored.
Web resource	R607

```
BEGIN TRANSACTION;
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

    --Update current role date
    UPDATE role
    SET endDate = now()
    WHERE id_user = $id;

   --Create new role
    INSERT INTO role(roleType, beginningDate, endDate , id_user)
    VALUES (moderator, now(), NULL, $id);

    COMMIT;
```

GROUP1834:

- João Augusto dos Santos Lima, up201605314@fe.up.pt
- José Diogo da Cunha Moreira Trindade Martins, up201504761@fe.up.pt
- Juliana Maria Cruz Marques, up201605568@fe.up.pt (Editor)
- Leonor Ribeiro e Sousa Mendes de Freitas, up201207603@fe.up.pt