# A9: Main accesses to the database and transactions- learnit (Q&A)

## 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.

## 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; | |

## 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; | |

## 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); | |

## 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) 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; | |

## 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); | |

## 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; | |

## REVISION HISTORY

Changes made to the first submission:

1. Fixed the query (SQL 503).

## 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