A9: Acessos Principais à Base de Dados e Transações

1 Acessos principais à base de dados

1.1 M01: Utilizadores

SQL01	Função que retorna informação importante acerca de um utilizador
Web Resource	R105

```
CREATE OR REPLACE FUNCTION user_profile(puser_id INT)
    RETURNS TABLE (
        fullname CHARACTER VARYING(200),
        username CHARACTER VARYING(50),
        email CHARACTER VARYING(70),
        about CHARACTER VARYING(200),
        location CHARACTER VARYING(100),
        ROLE CHARACTER VARYING(10),
    --badge character varying(50),
        created_at DATE,
        count_votes_rating_received INT,
        count_questions BIGINT,
        count_answers BIGINT,
        count_votes_made BIGINT
    ) AS $func$
BEGIN
    RETURN QUERY
    SELECT users.fullname, users.username, users.email, users.about,
        (SELECT locations.name FROM locations WHERE users.locationid =
locations.locationid),
        (SELECT name FROM users INNER JOIN userroles ON users.roleid =
userroles.roleid WHERE userid = puser_id),
        users.signup_date,
        count_vote_rating_received_user(puser_id),
        (SELECT COUNT(*) FROM publications INNER JOIN questions ON
questions.publicationid = publications.publicationid
            RIGHT JOIN users ON publications.userid = users.userid WHERE
users.userid = puser_id),
        (SELECT COUNT(*) FROM publications INNER JOIN answers ON
answers.publicationid = publications.publicationid
            RIGHT JOIN users ON publications.userid = users.userid WHERE
users.userid = puser_id),
        (SELECT COUNT(*) FROM votes WHERE votes.userid = puser_id)
    FROM users
    WHERE users.userid = puser_id;
END
$func$  LANGUAGE plpgsql;
```

SQL02	Função que actualiza o Badge de um utilizador
WebResource	R109

```
CREATE OR REPLACE FUNCTION user_badges_ranking()
    RETURNS TRIGGER AS $func$
DECLARE target_user INTEGER;
BEGIN
    SELECT publications.userid FROM publications INNER JOIN votes ON
publications.publicationid = votes.publicationid
    WHERE publications.publicationid = NEW.publicationid INTO target_user;
    IF count_vote_rating_received_user(target_user) = 1 THEN
        INSERT INTO userbadges(userid, badgeid) VALUES (target_user, 1);
    END IF;
    IF count_vote_rating_received_user(target_user) = 3 THEN
        INSERT INTO userbadges(userid, badgeid) VALUES (target_user, 2);
    END IF;
    IF count_vote_rating_received_user(target_user) = 15 THEN
        INSERT INTO userbadges(userid, badgeid) VALUES (target_user, 3);
    END IF;
    IF count_vote_rating_received_user(target_user) = 30 THEN
        INSERT INTO userbadges(userid, badgeid) VALUES (target_user, 4);
    END IF;
    IF count_vote_rating_received_user(target_user) = 50 THEN
        INSERT INTO userbadges(userid, badgeid) VALUES (target_user, 5);
    END IF;
    RETURN NULL;
END
$func$ LANGUAGE plpgsql;

CREATE TRIGGER auto_rank_up AFTER INSERT OR UPDATE ON votes
FOR EACH ROW EXECUTE PROCEDURE user_badges_ranking();
```

SQL03	Função que conta os votos que um utilizador recebeu numa determinada publicação

```
CREATE OR REPLACE FUNCTION count_vote_rating_received_user(puser_id INT)
    RETURNS INTEGER AS $func$
DECLARE publicationvotecount INTEGER;
BEGIN
    SELECT COUNT(*) FROM votes INNER JOIN publications ON
votes.publicationid = publications.publicationid
        RIGHT JOIN users ON publications.userid = users.userid WHERE
users.userid = puser_id
    INTO publicationvotecount;

    IF publicationvotecount IS NULL THEN
        publicationvotecount := ;
    END IF;

    RETURN publicationvotecount;
END
$func$  LANGUAGE plpgsql;
```

SQL04	Função que retorna os melhores utilizadores
WebResource	R401

```
CREATE OR REPLACE FUNCTION top_scored_users()
  RETURNS TABLE (
    username CHARACTER VARYING(50),
  --badge character varying(50),
    count_votes_rating_received INT,
```

```
    count_questions INT,
    count_answers INT,
    count_comments INT
  ) AS $func$
BEGIN
  RETURN QUERY
  SELECT users.username,
    count_vote_rating_received_user(users.userid) AS total_votes,
    user_total_questions(users.userid) AS total_questions,
    user_total_answers(users.userid) AS total_answers,
    user_total_comments(users.userid) AS total_comments
  FROM votes
    INNER JOIN publications
      ON votes.publicationid = publications.publicationid
    INNER JOIN users
      ON publications.userid = users.userid
  GROUP BY users.userid
  ORDER BY total_votes
  DESC LIMIT 5;
END
$func$  LANGUAGE plpgsql;
```

SQL05	Funções que retornam valores das estatísticas dos utilizadores
WebResource	R401, R105

```
CREATE OR REPLACE FUNCTION user_total_questions(puser_id INT)
  RETURNS INTEGER
LANGUAGE plpgsql
AS $$
DECLARE questions_count INTEGER;
BEGIN
  SELECT COUNT(*) FROM questions
    INNER JOIN publications
      ON questions.publicationid = publications.publicationid
  WHERE publications.userid = puser_id
  INTO questions_count;

  IF questions_count IS NULL THEN
    questions_count := ;
  END IF;

  RETURN questions_count;
END
$$;

CREATE OR REPLACE FUNCTION user_total_answers(puser_id INT)
  RETURNS INTEGER
LANGUAGE plpgsql
AS $$
DECLARE questions_count INTEGER;
BEGIN
```

```
  SELECT COUNT(*) FROM answers
    INNER JOIN publications
      ON answers.publicationid = publications.publicationid
  WHERE publications.userid = puser_id
  INTO questions_count;

  IF questions_count IS NULL THEN
    questions_count := ;
  END IF;

  RETURN questions_count;
END
$$;

CREATE OR REPLACE FUNCTION user_total_comments(puser_id INT)
  RETURNS INTEGER
LANGUAGE plpgsql
AS $$
DECLARE questions_count INTEGER;
BEGIN
  SELECT COUNT(*) FROM comments
    INNER JOIN publications
      ON comments.publicationid = publications.publicationid
  WHERE publications.userid = puser_id
  INTO questions_count;

  IF questions_count IS NULL THEN
    questions_count := ;
  END IF;

  RETURN questions_count;
END
$$;
```

1.2 M02: Perguntas

SQL04	Função que retorna as tags de uma pergunta
WebResource	R205

```
CREATE OR REPLACE FUNCTION question_tags(pquestion_id INT)
    RETURNS TABLE (tag CHARACTER VARYING(10)) AS $func$
BEGIN
    RETURN QUERY
    SELECT tags.name
    FROM tags INNER JOIN questiontags ON tags.tagid = questiontags.tagid
    WHERE questiontags.questionid = pquestion_id;
END
$func$  LANGUAGE plpgsql;
```

SQL05	Função que retorna as respostas a uma pergunta e informação adicional

```
CREATE OR REPLACE FUNCTION question_answers(pquestion_id INT)
    RETURNS TABLE (
        id INTEGER,
        user_id INTEGER,
        username CHARACTER VARYING(50),
        body TEXT,
        created_at TIMESTAMP
    ) AS $func$
BEGIN
    RETURN QUERY
    SELECT answers.publicationid, users.userid, users.username,
publications.body, publications.creation_date
    FROM answers INNER JOIN publications ON answers.publicationid =
publications.publicationid
        RIGHT JOIN users ON publications.userid = users.userid
    WHERE answers.questionid = pquestion_id;
END
$func$  LANGUAGE plpgsql;
```

SQL06	Função que actualiza a data de última modificação

```
DROP TRIGGER IF EXISTS answer_update_question_timestamp ON
public.publications;

CREATE OR REPLACE FUNCTION trigger_update_question_timestamp()
    RETURNS TRIGGER AS $func$
BEGIN
    NEW.last_edit_date := now();
    RETURN NEW;
END;
$func$  LANGUAGE plpgsql;

CREATE TRIGGER answer_update_question_timestamp BEFORE INSERT OR UPDATE ON
publications
FOR EACH ROW EXECUTE PROCEDURE trigger_update_question_timestamp();
```

SQL07	Função que retorna o username do autor de uma pergunta

```
CREATE OR REPLACE FUNCTION getusernamefromquestion(questionid INTEGER)
    RETURNS VARCHAR AS $$
BEGIN
    SELECT users.username
    FROM publications
        INNER JOIN questions ON publications.publicationid =
questions.publicationid
        INNER JOIN users ON publications.userid = users.userid
    WHERE questions.publicationid = $1;
END;
$$ LANGUAGE plpgsql;
```

SQL08	Função que retorna os detalhes das perguntas mais votadas
WebResource	R208

```
CREATE OR REPLACE FUNCTION top_scored_questions(skip INTEGER, limitnumber
INTEGER)
    RETURNS
        TABLE
        (
            publicationid INTEGER,
            title CHARACTER VARYING,
            body text,
            creation_date TIMESTAMP WITHOUT TIME zone,
            solved_date TIMESTAMP WITHOUT TIME zone,
            username CHARACTER VARYING,
            userid INTEGER,
            answers_count BIGINT,
            upvotes BIGINT,
            votes_count BIGINT,
            views_counter BIGINT)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT questions.publicationid, questions.title, publications.body,
        publications.creation_date, questions.solved_date, users.username,
users.userid,
        (SELECT COUNT(*) FROM question_answers(questions.publicationid)) AS
answers_count,
        (SELECT COUNT (*) FROM votes WHERE votes.values = 1 AND
votes.publicationid = 1) AS upvotes,
        (SELECT COALESCE(SUM(votes.values), ) FROM votes WHERE
votes.publicationid = questions.publicationid) AS votes_count,
        questions.views_counter
    FROM questions
        INNER JOIN publications
            ON questions.publicationid = publications.publicationid
        LEFT JOIN users ON publications.userid = users.userid
    ORDER BY votes_count DESC
    LIMIT limitNumber
    OFFSET skip;
END
$$;
```

SQL09	Função que retorna os detalhes de uma pergunta
WebResource	R205

```
CREATE OR REPLACE FUNCTION question_details_from_id (pubid INTEGER)
    RETURNS TABLE (
        publicationid INTEGER,
        title VARCHAR(100),
        body TEXT,
        creation_date TIMESTAMP,
        solved_date TIMESTAMP,
        username VARCHAR(10),
        userid INTEGER,
        answers_count BIGINT,
        upvotes BIGINT,
        down_votes BIGINT,
        views_counter BIGINT,
        category VARCHAR(100))
AS $func$
BEGIN
    RETURN QUERY
    SELECT questions.publicationid,  questions.title, publications.body,
        publications.creation_date, questions.solved_date,
users.username,users.userid,
        (SELECT COUNT(*) FROM question_answers(questions.publicationid)) AS
answers_count,
        (SELECT COUNT (*) FROM votes WHERE votes.publicationid = pubid AND
votes.values = 1) AS upvotes,
        (SELECT COUNT (*) FROM votes WHERE votes.publicationid = pubid AND
votes.values = -1) AS down_votes,
        questions.views_counter,
        categories.name
    FROM questions
        INNER JOIN publications
            ON questions.publicationid = publications.publicationid
        INNER JOIN categories ON questions.categoryid =
categories.categoryid
        LEFT JOIN users ON publications.userid = users.userid
    WHERE questions.publicationid = pubid;
END
$func$  LANGUAGE plpgsql;
```

SQL10	Função que retorna os detalhes das perguntas mais recentes
WebResource	R207

```
CREATE OR REPLACE FUNCTION recent_questions(skip INTEGER, limitNumber
INTEGER)
    RETURNS TABLE (
        publicationid INTEGER,
        title VARCHAR(100),
        body TEXT,
        creation_date TIMESTAMP,
        solved_date TIMESTAMP,
        username VARCHAR(10),
        userid INTEGER,
        answers_count BIGINT,
        upvotes BIGINT,
        votes_count BIGINT,
        views_counter BIGINT)
AS $func$
BEGIN
    RETURN QUERY
    SELECT questions.publicationid, questions.title, publications.body,
        publications.creation_date, questions.solved_date, users.username,
users.userid,
        (SELECT COUNT(*) FROM question_answers(questions.publicationid)) AS
answers_count,
        (SELECT COUNT (*) FROM votes WHERE votes.values = 1 AND
votes.publicationid = 1) AS upvotes,
        (SELECT SUM(votes.values) FROM votes WHERE votes.publicationid =
questions.publicationid),
        questions.views_counter
    FROM questions
        INNER JOIN publications
            ON questions.publicationid = publications.publicationid
        LEFT JOIN users ON publications.userid = users.userid
    ORDER BY creation_date DESC
    LIMIT limitNumber
    OFFSET skip;
END
$func$  LANGUAGE plpgsql;
```

SQL11	Função que retorna os detalhes das perguntas não respondidas
WebResource	R209

```
CREATE OR REPLACE FUNCTION unanswered_questions(skip INTEGER, limitnumber
INTEGER)
    RETURNS TABLE
    (
        publicationid INTEGER,
        title CHARACTER VARYING,
        body text,
        creation_date TIMESTAMP WITHOUT TIME zone,
        solved_date TIMESTAMP WITHOUT TIME zone,
        username CHARACTER VARYING,
        userid INTEGER,
        answers_count BIGINT,
        upvotes BIGINT,
        votes_count BIGINT,
        views_counter BIGINT)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT questions.publicationid, questions.title, publications.body,
        publications.creation_date, questions.solved_date, users.username,
users.userid,
        (SELECT COUNT(*) FROM question_answers(questions.publicationid)) AS
answers_count,
        (SELECT COUNT (*) FROM votes WHERE votes.values = 1 AND
votes.publicationid = 1) AS upvotes,
        (SELECT SUM(votes.values) FROM votes WHERE votes.publicationid =
questions.publicationid) AS votes_count,
        questions.views_counter
    FROM questions
        INNER JOIN publications
            ON questions.publicationid = publications.publicationid
        LEFT JOIN users ON publications.userid = users.userid
    LIMIT limitNumber
    OFFSET skip;
END
$$;
```

SQL12	Função que retorna as respostas de uma pergunta
WebResource	R205

```
CREATE OR REPLACE FUNCTION answers_from_questionid(qid INTEGER)
    RETURNS TABLE (
        answerid INTEGER,
        body TEXT,
        solved_date TIMESTAMP,
        creation_date TIMESTAMP,
        userid INTEGER,
        username VARCHAR(50)
    )
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT answers.publicationid, publications.body, answers.solved_date,
publications.creation_date, users.userid, users.username
    FROM publications INNER JOIN answers ON publications.publicationid =
answers.publicationid
        LEFT JOIN users ON publications.userid = users.userid
    WHERE answers.questionid = qid;
END
$$;
```

SQL13	Função para editar uma pergunta
WebResource	R204

```
CREATE OR REPLACE FUNCTION update_question(body_edited text, questionid
INTEGER, title_edited VARCHAR, categoryid_edited INTEGER)
    RETURNS void LANGUAGE plpgsql AS $$
DECLARE RESULT INTEGER;
BEGIN
    UPDATE publications
    SET body = body_edited
    WHERE publications.publicationid = questionid
    returning publications.publicationid AS publicationid INTO RESULT;

    UPDATE questions
    SET title = title_edited, categoryid = categoryid_edited
    WHERE questions.publicationid = questionid;
END $$;
```

SQL14	Função para criar um comentário

```
CREATE OR REPLACE FUNCTION insert_into_answercomments(userid INTEGER,
answerid INTEGER, body text)
    RETURNS void LANGUAGE plpgsql AS $$
DECLARE RESULT INTEGER;
BEGIN
    INSERT INTO publications(body, userid)
    VALUES (body, userid)
    returning publications.publicationid AS publicationid INTO RESULT;

    INSERT INTO comments(publicationid) VALUES (RESULT);
    INSERT INTO answercomments(commentid, answerid) VALUES (RESULT,
answerid);
END $$;
```

SQL15	Função que verifica se o utilizador está a votar no próprio conteúdo
WebResource	R214

```
CREATE OR REPLACE FUNCTION own_content_vote()
    RETURNS TRIGGER AS $func$
DECLARE target_user INTEGER;
BEGIN
    SELECT publications.userid FROM publications INNER JOIN votes ON
publications.publicationid = votes.publicationid
    WHERE publications.publicationid = NEW.publicationid INTO target_user;
    IF target_user = NEW.userid THEN
        RAISE EXCEPTION 'You cant vote on your own publications';
    END IF;
    RETURN NULL;
END
$func$  LANGUAGE plpgsql;

CREATE TRIGGER own_content_vote_trigger AFTER INSERT OR UPDATE ON votes
FOR EACH ROW EXECUTE PROCEDURE own_content_vote();
```

SQL16	Função que retorna os detalhes das perguntas de um utilizador
Web Resource	R105

```
CREATE OR REPLACE FUNCTION get_questions_by_user_id (uid INTEGER, skip
INTEGER, limitnumber INTEGER)
    RETURNS TABLE (
        publicationid INTEGER,
        title VARCHAR(100),
        body TEXT,
        creation_date TIMESTAMP,
        solved_date TIMESTAMP,
        username VARCHAR(10),
        userid INTEGER,
        answers_count BIGINT,
        upvotes BIGINT,
        votes_count BIGINT,
        views_counter BIGINT)
LANGUAGE plpgsql
AS $func$
BEGIN
    RETURN QUERY
    SELECT questions.publicationid, questions.title, publications.body,
        publications.creation_date, questions.solved_date, users.username,
users.userid,
        (SELECT COUNT(*) FROM question_answers(questions.publicationid)) AS
answers_count,
        (SELECT COUNT (*) FROM votes WHERE votes.values = 1 AND
votes.publicationid = 1) AS upvotes,
        (SELECT SUM(votes.values) FROM votes WHERE votes.publicationid =
questions.publicationid) AS votes_count,
        questions.views_counter
    FROM questions
        INNER JOIN publications
            ON questions.publicationid = publications.publicationid
        LEFT JOIN users ON publications.userid = users.userid
    WHERE users.userid = uid
    LIMIT limitNumber
    OFFSET skip;
END
$func$;
```

SQL17	Função para pesquisa de texto em publicações
WebResource	R206

```
CREATE OR REPLACE FUNCTION search_questions(psearch text)
    RETURNS TABLE (questionid INTEGER) AS $func$
BEGIN
    RETURN QUERY
    SELECT DISTINCT publications.publicationid
    FROM questions, publications
    WHERE to_tsvector(COALESCE(questions.title,'') || ' ' ||
COALESCE(publications.body,'')) @@ to_tsquery(psearch)
          OR questions.publicationid IN (
        SELECT DISTINCT(answers.questionid) FROM answers INNER JOIN
publications ON answers.publicationid = publications.publicationid
        WHERE to_tsvector(COALESCE(publications.body)) @@
to_tsquery(psearch)
    )
    ;
END
$func$  LANGUAGE plpgsql;
```

SQL18	Função que retorna os comentários de uma resposta
WebResource	R205

```
CREATE OR REPLACE FUNCTION get_answer_comments (aid INTEGER)
  RETURNS TABLE (
    publicationid INTEGER,
    body TEXT,
    creation_date TIMESTAMP,
    userid INTEGER,
    username VARCHAR(50)

  )
LANGUAGE plpgsql
AS $$
BEGIN
  RETURN QUERY
  SELECT publications.publicationid, publications.body,
publications.creation_date, users.userid, users.username
  FROM answercomments INNER JOIN publications ON answercomments.commentid =
publications.publicationid
    LEFT JOIN users ON publications.userid = users.userid
  WHERE answercomments.answerid = aid;
END
$$;
```

SQL19	Função que retorna os comentários de uma pergunta
WebResource	R205

```
CREATE OR REPLACE FUNCTION get_question_comments (qid INTEGER)
  RETURNS TABLE (
    publicationid INTEGER,
    body TEXT,
    creation_date TIMESTAMP,
    userid INTEGER,
    username VARCHAR(50)
```

```
  )
LANGUAGE plpgsql
AS $$
BEGIN
  RETURN QUERY
  SELECT publications.publicationid, publications.body,
publications.creation_date, users.userid, users.username
  FROM questioncomments INNER JOIN publications ON
questioncomments.commentid = publications.publicationid
    LEFT JOIN users ON publications.userid = users.userid
  WHERE questioncomments.questionid= qid;
END
$$;
```

1.3 M03: Admin

SQL18	Função para banir um utilizador quando este excede o limite de warnings
Web Resource	R307

```
DROP TRIGGER IF EXISTS auto_ban_on_warning_limit ON public.warnings;

CREATE OR REPLACE FUNCTION trigger_auto_ban_on_warning_limit()
    RETURNS "trigger" AS $func$
BEGIN
    IF (SELECT COUNT(*)
        FROM modregisters INNER JOIN users ON modregisters.userid_author =
users.userid
            INNER JOIN warnings ON modregisters.modregisterid =
warnings.warningid
        GROUP BY userid_target) = 3 THEN
        INSERT INTO bans(banid) VALUES(NEW.warningid);
    END IF;
    RETURN NULL;
END;
$func$  LANGUAGE plpgsql;

CREATE TRIGGER auto_ban_on_warning_limit AFTER INSERT ON warnings
FOR EACH ROW EXECUTE PROCEDURE trigger_auto_ban_on_warning_limit();
```

2 Transações

SQL01	Create a comment for a specific answer
Isolation Level	READ COMMITTED

```
BEGIN;
INSERT INTO publications(body, userid)
VALUES (?, ?)
RETURNING publications.publicationid AS publicationid INTO RESULT;

INSERT INTO comments(publicationid) VALUES (RESULT);
```

```
INSERT INTO answercomments(commentid, answerid) VALUES (RESULT, ?);
COMMIT;
```

SQL02	Create a comment for a specific question
Isolation Level	READ COMMITTED

```
BEGIN;
INSERT INTO publications(body, userid)
VALUES (?, ?)
RETURNING publications.publicationid AS publicationid INTO RESULT;

INSERT INTO comments(publicationid) VALUES (RESULT);

INSERT INTO questioncomments(commentid, questionid) VALUES (RESULT, ?);
COMMIT;
```

SQL03	Create an answer for a specific question
Isolation Level	READ COMMITTED

```
BEGIN;
INSERT INTO publications(body, userid)
VALUES (body, userid)
returning publications.publicationid AS publicationid INTO RESULT;

INSERT INTO answers(publicationid , questionid) VALUES (RESULT, questionid);
COMMIT;
```

SQL04	Create a question
Isolation Level	READ COMMITTED

```
BEGIN;
INSERT INTO publications(body, userid)
VALUES (body, userid)
returning publications.publicationid AS publicationid INTO RESULT;

INSERT INTO questions(publicationid ,title, categoryid) VALUES (RESULT,
title, categoryid);
COMMIT;
```

SQL05	Edit Question
Isolation Level	READ COMMITTED

```
BEGIN;
    UPDATE publications
    SET body = ?
    WHERE publications.publicationid = questionid
    returning publications.publicationid AS publicationid INTO RESULT;

    UPDATE questions
    SET title = title_edited, categoryid = categoryid_edited
    WHERE questions.publicationid = questionid;
```

```
COMMIT;
```

SQL06	Warn User
Isolation Level	READ COMMITTED

```
BEGIN;
INSERT INTO modegisters(reason, userid_author, userid_target)
VALUES (?, ?, ?)
returning publications.publicationid AS publicationid INTO RESULT;

INSERT INTO warnings(publicationid) VALUES (RESULT);
COMMIT;
```

SQL07	Ban a User
Isolation Level	READ COMMITTED

```
BEGIN;
INSERT INTO modegisters(reason, userid_author, userid_target)
VALUES (?, ?, ?)
returning publications.publicationid AS publicationid INTO RESULT;

INSERT INTO bans(publicationid, end_date) VALUES (RESULT, ?);
COMMIT;
```

Revisão

- Badges - ON CONFLICT DO NOTHING
- SQL05 - removido returning
- Transacções - ajustado nível de isolamento para READ UNCOMMITED