**A6 – Integrity constraints. Indexes, triggers, user functions and database populated with data**

1. **Database workload**
   1. **Estimate of tuples**

|  |  |  |  |
| --- | --- | --- | --- |
| Relation reference | Relation name | Order of magnitude | Estimated growth |
| R01 | User | Dozens | Units per week |
| R02 | Project | Hundreds | Dozens per month |
| R03 | Board | Hundreds | Dozens per month |
| R04 | Task | Thousands | Dozens per week |
| R05 | Meeting | Hundreds | Units per week |
| R06 | Personal\_event | Hundreds | Units per week |
| R07 | Comment | Dozens | Units per week |
| R08 | File | Dozens | Units per week |
| R09 | Notification | Thousands | Units per day |
| R10 | Progress\_update | Thousands | Units per day |
| R11 | Message | Hundreds | Units per week |
| R12 | Project\_team | Hundreds | Dozens per month |
| R13 | Board\_team | Hundreds | Dozens per month |
| R14 | Profile\_picture | Dozens | No growth |
| R15 | Project\_picture | Hundreds | No growth |
| R16 | Contact | Dozens | Units per month |
| R17 | Ban\_user | Units | Units per year |
| R18 | Archive\_project | Dozens | Units per month |

* 1. **Most frequent queries**

|  |  |
| --- | --- |
| Query reference | SELECT01 |
| Query description | All emails |
| Query frequency |  |
| SELECT e\_mail FROM "User" | |

|  |  |
| --- | --- |
| Query reference | SELECT02 |
| Query description | Email’s password |
| Query frequency |  |
|  | |
| SELECT password FROM "User"  WHERE user.e\_mail = $userEmail | |

|  |  |
| --- | --- |
| Query reference | SELECT03 |
| Query description | All public projects |
| Query frequency |  |
| SELECT name FROM Project  WHERE Project.privacy=1 | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT04 |
| Query description | All user’s projects |
| Query frequency |  |
| SELECT \*  FROM Project  INNER JOIN Project\_team ON Project.id=Project\_team.id\_project  WHERE Project.id\_coordinator=$user OR Project\_team.id\_user=$user | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT05 |
| Query description | All user’s personal events |
| Query frequency |  |
| SELECT \* FROM Personal\_event  WHERE Personal\_event.id\_user=$user | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT06 |
| Query description | All users board’s meetings |
| Query frequency |  |
| SELECT \*  FROM Meeting  INNER JOIN Board\_team ON Meeting.id\_board=Board\_team.id\_board  WHERE Board\_team.id\_worker=$user | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT07 |
| Query description | User’s personal info |
| Query frequency |  |
| SELECT \* FROM "User"  WHERE User.id=$user | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT08 |
| Query description | User’s notifications |
| Query frequency |  |
| SELECT \* FROM Notification  WHERE Notification.id\_user=$user | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT09 |
| Query description | User’s number of unread notifications |
| Query frequency |  |
| SELECT COUNT id  FROM Notification  WHERE Notification.id\_user=$user AND Notification.read=0 | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT10 |
| Query description | Project’s boards |
| Query frequency |  |
| SELECT \*  FROM Board  WHERE Board.id\_project=$projectId | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT11 |
| Query description | Project’s info |
| Query frequency |  |
| SELECT \* FROM Project  WHERE Project.id=$projectId | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT12 |
| Query description | Project worker’s info |
| Query frequency |  |
| SELECT User.full\_name, Profile\_picture.path, User.e\_mail  FROM "User"  INNER JOIN Profile\_picture ON Profile\_picture.id\_user=$user  WHERE Project.id=$projectId AND Profile\_picture.path=SELECT(MAX(Profile\_picture.id)) | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT13 |
| Query description | Project’s events |
| Query frequency |  |
| SELECT \*  FROM Meeting  INNER JOIN Board ON Board.id=Meeting.id\_board  WHERE Board.id\_project=$projectId | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT14 |
| Query description | Last week project’s messages |
| Query frequency |  |
| SELECT \*  FROM Message  WHERE Message.id\_project=$projectId AND Message.date >= DATEADD(day, -7, GETDATE()) | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT15 |
| Query description | Board’s tasks |
| Query frequency |  |
| SELECT Task.id  FROM Task  WHERE Task.id\_board=$boardId | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT16 |
| Query description | Task’s info |
| Query frequency |  |
| SELECT \*  FROM Task  WHERE Task.id=$taskId | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT15 |
| Query description | Last 5 task’s updates |
| Query frequency |  |
| SELECT TOP 5 Comment.id, File.id, Progress\_update.id  FROM Task  INNER JOIN Comment ON Task.id=Comment.id\_task  INNER JOIN File ON Task.id=File.id\_task  INNER JOIN Task.id=Progress\_update.id\_task  WHERE Task.id=$taskId | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT16 |
| Query description | Search project |
| Query frequency |  |
| SELECT description, start\_date, name FROM Project  WHERE description LIKE %$search% OR title LIKE %$search% AND privacy=false;  ORDER BY title; | |
|  | |

|  |  |
| --- | --- |
| Query reference | SELECT17 |
| Query description | Search user |
| Query frequency |  |
| SELECT username, full\_name, e\_mail FROM User  WHERE username LIKE %$search% OR full\_name LIKE %$search%;  ORDER BY username; | |
|  | |

* 1. **Most frequent modifications**

|  |  |
| --- | --- |
| Query reference | UPDATE01 |
| Query description | Insert new progress update |
| Query frequency |  |
| INSERT INTO Progress\_update (new\_value, id\_user, id\_task)  VALUES ($new\_value, $id\_user, $id\_task) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE02 |
| Query description | Update task progress |
| Query frequency |  |
| UPDATE Task  SET progress = $progress | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE03 |
| Query description | Upload new file |
| Query frequency |  |
| INSERT INTO File (path, description, id\_user, id\_task)  VALUES ($path, $description, $id\_user, $id\_task) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE04 |
| Query description | Comment task |
| Query frequency |  |
| INSERT INTO Comment (comment, id\_user, id\_task)  VALUES ($comment, $id\_user, $id\_task) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE05 |
| Query description | Add personal event |
| Query frequency |  |
| INSERT INTO Personal\_event (date, place, name, id\_user)  VALUES ($date, $place, $name, $id\_user) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE06 |
| Query description | Add board meeting |
| Query frequency |  |
| INSERT INTO Meeting (date, name, place, id\_board)  VALUES ($date, $name, $place, $id\_board) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE07 |
| Query description | Add new project |
| Query frequency |  |
| INSERT INTO Project (description, name, id\_coordinator, privacy)  VALUES ($description, $name, $id\_coordinator, $privacy) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE08 |
| Query description | Add new board to existing project |
| Query frequency |  |
| INSERT INTO Board (description, name, id\_creator, id\_project)  VALUES ($description, $name, $id\_creator, $id\_project) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE09 |
| Query description | Archive or close task |
| Query frequency |  |
| UPDATE Task  SET task\_state=$task\_state | |
|  | |
| Query reference | UPDATE10 |
| Query description | Archive or close project |
| Query frequency |  |
| UPDATE Project  SET project\_state=$project\_state | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE11 |
| Query description | Add new worker to project |
| Query frequency |  |
| INSERT INTO Project\_team (id\_user, id\_project)  VALUES ($id\_user, $id\_project) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE12 |
| Query description | Add new worker to board |
| Query frequency |  |
| INSERT INTO Board\_team (id\_board, id\_user)  VALUES ($id\_board, $id\_user) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE13 |
| Query description | Add new user |
| Query frequency |  |
| INSERT INTO User (e\_mail, password, URL, username)  VALUES ($e\_mail, $password, $URL, $username) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE14 |
| Query description | Update user’s full name |
| Query frequency |  |
| UPDATE User  SET full\_name=$full\_name | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE15 |
| Query description | Change password |
| Query frequency |  |
| UPDATE User  SET password=$password | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE16 |
| Query description | Set user as administrator |
| Query frequency |  |
| UPDATE User  SET administrator=$administrator | |
|  | |
| Query reference | UPDATE17 |
| Query description | Change profile picture |
| Query frequency |  |
| INSERT INTO Profile\_picture (id\_user, path)  VALUES ($id\_user, $path) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE18 |
| Query description | Change project description |
| Query frequency |  |
| UPDATE Project  SET description=$description | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE19 |
| Query description | Add project end date |
| Query frequency |  |
| UPDATE Project  SET end\_date=$end\_date | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE20 |
| Query description | Change project picture |
| Query frequency |  |
| INSERT INTO Project\_picture (id\_project, path)  VALUES ($id\_project, $path) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE21 |
| Query description | Add user to contacts |
| Query frequency |  |
| INSERT INTO Contact (id\_user, id\_contact)  VALUES ($id\_user, $id\_contact) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE22 |
| Query description | Send messages to project forum |
| Query frequency |  |
| INSERT INTO Message (message, id\_user, id\_project)  VALUES ($message, $id\_user, $id\_project) | |
|  | |

|  |  |
| --- | --- |
| Query reference | UPDATE23 |
| Query description | Create new notification |
| Query frequency |  |
| INSERT INTO Notification (id\_user, notification)  VALUES ($id\_user, $notification) | |
|  | |
| Query reference | UPDATE24 |
| Query description | Mark notification as read or unread |
| Query frequency |  |
| UPDATE Notification  SET read=$read | |
|  | |

1. **Proposed Indexes**
   1. **Performance indexes**

|  |  |
| --- | --- |
| Index reference | INDEX01 |
| Related queries | SELECT07 |
| Index relation | User |
| Index attribute | e\_mail |
| Index type | Hash |
| Cardinality | High |
| Clustering | No |
| Justification | Query needs to be executed many times and has to be fast; cardinality is high because the e\_mail is a unique key; not a good candidate for clustering. |
| CREATE INDEX email\_user ON User USING hash(e\_mail) | |
|  | |

|  |  |
| --- | --- |
| Index reference | INDEX02 |
| Related queries | SELECT03 |
| Index relation | Project |
| Index attribute | id |
| Index type | Hash |
| Cardinality | High |
| Clustering | No |
| Justification | Query needs to be executed many times and has to be fast; cardinality is high because the id is a unique key; not a good candidate for clustering. |
| CREATE INDEX id\_project ON Project USING hash(id) | |
|  | |

|  |  |
| --- | --- |
| Index reference | INDEX03 |
| Related queries | SELECT04 |
| Index relation | Project\_team |
| Index attribute | id\_user |
| Index type | Hash |
| Cardinality | Medium |
| Clustering | Yes |
| Justification | Query needs to be executed many times and table is very large, so it has be fast; cardinality is medium so it’s a good candidate for clustering. |
| CREATE INDEX user\_project ON Project\_team USING hash(id\_user) | |
|  | |

|  |  |
| --- | --- |
| Index reference | INDEX04 |
| Related queries | SELECT10 |
| Index relation | Board |
| Index attribute | id\_project |
| Index type | Hash |
| Cardinality | Medium |
| Clustering | Yes |
| Justification | Query needs to be executed many times and has to be fast because it is executed many times; cardinality is medium so it’s a good candidate for clustering. |
| CREATE INDEX project\_boards ON Board USING hash(id\_project) | |
|  | |

|  |  |
| --- | --- |
| Index reference | INDEX05 |
| Related queries | SELECT14 |
| Index relation | Message |
| Index attribute | date, project |
| Index type | B-tree |
| Cardinality | Medium |
| Clustering | Yes |
| Justification | Query needs to be executed many times and table is very large; cardinality is medium so it’s a good candidate for clustering. |
| CREATE INDEX message\_date ON Message USING btree(date, id\_project) | |
|  | |

* 1. **Full-text Search indexes**

|  |  |
| --- | --- |
| Index reference | INDEX06 |
| Related queries | SELECT16 |
| Index relation | Project |
| Index attribute | title |
| Index type | GiST |
| Clustering | No |
| Justification | To improve the performance of full text searches while searching for projects by their titles; GiST because it’s better for dynamic data. |
| CREATE INDEX search\_project ON Project USING GIST(title) | |
|  | |

|  |  |
| --- | --- |
| Index reference | INDEX07 |
| Related queries | SELECT17 |
| Index relation | User |
| Index attribute | username |
| Index type | GiST |
| Clustering | No |
| Justification | To improve the performance of full text searches while searching for users by their username; GiST because it’s better for dynamic data. |
| CREATE INDEX search\_user ON User USING GIST(username) | |
|  | |

1. **Triggers**

|  |  |
| --- | --- |
| Trigger reference | TRIGGER01 |
| Trigger description | An e-mail can only be registered once |
| |  | | --- | | CREATE FUNCTION check\_exist\_email() RETURNS TRIGGER AS | |  | $BODY$ | |  | BEGIN | |  | IF EXISTS (SELECT e\_mail FROM "User" WHERE NEW.e\_mail = e\_mail) THEN | |  | RAISE EXCEPTION 'The e-mail you are trying to use is already registred' | |  | END IF; | |  | RETURN NEW; | |  | END | |  | $BODY$ | |  | LANGUAGE plpgsql; | |  |  | |  | CREATE TRIGGER check\_exist\_email | |  | BEFORE INSERT ON "User" | |  | FOR EACH ROW | |  | EXECUTE PROCEDURE check\_exist\_user(); | | |

|  |  |
| --- | --- |
| Trigger reference | TRIGGER02 |
| Trigger description | An username can only be registred once |
| |  | | --- | | CREATE FUNCTION check\_exist\_username() RETURNS TRIGGER AS | |  | $BODY$ | |  | BEGIN | |  | IF EXISTS (SELECT username FROM "User" WHERE NEW.username = username) THEN | |  | RAISE EXCEPTION 'The username you are trying to use is already in use' | |  | END IF; | |  | RETURN NEW; | |  | END | |  | $BODY$ | |  | LANGUAGE plpgsql; | |  |  | |  | CREATE TRIGGER check\_exist\_username | |  | BEFORE INSERT ON "User" | |  | FOR EACH ROW | |  | EXECUTE PROCEDURE check\_exist\_username(); | | |

|  |  |
| --- | --- |
| Trigger reference | TRIGGER03 |
| Trigger description | Two boards of the same project can’t have the same name |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | CREATE FUNCTION check\_exist\_board() RETURNS TRIGGER AS | |  | $BODY$ | |  | BEGIN | |  | IF EXISTS (SELECT name FROM Board WHERE Board.id\_project=$projectId AND NEW.name=name) THEN | |  | RAISE EXCEPTION 'This project already has a board with that name' | |  | END IF; | |  | RETURN NEW; | |  | END | |  | $BODY$ | |  | LANGUAGE plpgsql; | |  |  | |  | CREATE TRIGGER check\_exist\_board | |  | BEFORE INSERT ON Board | |  | FOR EACH ROW | |  | EXECUTE PROCEDURE check\_exist\_board(); | | |  | |  | | |

|  |  |
| --- | --- |
| Trigger reference | TRIGGER04 |
| Trigger description | Can’t add user to team if he already belongs to it |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | |  | | |  | | --- | | CREATE FUNCTION check\_exist\_worker() RETURNS TRIGGER AS | |  | $BODY$ | |  | BEGIN | |  | IF EXISTS (SELECT name FROM Project\_team WHERE Project\_team.id\_project=$projectId AND NEW.id\_worker=id\_worker) THEN | |  | RAISE EXCEPTION 'This user already belongs to this team' | |  | END IF; | |  | RETURN NEW; | |  | END | |  | $BODY$ | |  | LANGUAGE plpgsql; | |  |  | |  | CREATE TRIGGER check\_exist\_worker | |  | BEFORE INSERT ON Project\_team | |  | FOR EACH ROW | |  | EXECUTE PROCEDURE check\_exist\_worker(); | | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | | |

|  |  |
| --- | --- |
| Trigger reference | TRIGGER05 |
| Trigger description | Can’t add user to contact list if he already belongs to it |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | CREATE FUNCTION check\_exist\_contact() RETURNS TRIGGER AS | |  | $BODY$ | |  | BEGIN | |  | IF EXISTS (SELECT name FROM Contact WHERE Contact.id\_user=$userId AND NEW.id\_contact=id\_contact AND NEW.id\_contact=$userId) THEN | |  | RAISE EXCEPTION 'This user already is on your contact list' | |  | END IF; | |  | RETURN NEW; | |  | END | |  | $BODY$ | |  | LANGUAGE plpgsql; | |  |  | |  | CREATE TRIGGER check\_exist\_contact | |  | BEFORE INSERT ON Contact | |  | FOR EACH ROW | |  | EXECUTE PROCEDURE check\_exist\_contact(); | |  | |  |  | | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | | |

1. **SQL Code**