

# A9: Main accesses to the database and transactions

## 1. Main Accesses

Main accesses to the database.

### 1.1 M01 Register User

SQL Reference	Register User
Web Resource	R104

```sql

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

|                        |                                                           |
|------------------------|-----------------------------------------------------------|
| ### 1.2 M02 Feed Posts |                                                           |
| SQL Reference          | Get Posts from bands and musicians that user is following |
| -----                  | -----                                                     |
| Web Resource           | R301                                                      |

sql

```
SELECT post_id,poster_id,posterName,bandName,date, commentId
FROM
```

```
(SELECT user_follower.followingUserId as follower, post.private as private, post.id
      content.date as date, comment.id as commentId--, content.text as postText
FROM post
JOIN content ON content.id = post.contentId
JOIN mb_user ON mb_user.id = content.creatorId
JOIN user_follower ON user_follower.followedUserId = content.creatorId
LEFT JOIN comment ON comment.postId = post.id
LEFT JOIN content as commentContent ON commentContent.id = comment.contentId
LEFT JOIN mb_user as commentUser ON commentContent.creatorId = commentUser.id
```

UNION ALL

```
SELECT band_follower.userId as follower, post.private as private, post.id as post_id,
       content.date as date, comment.id as commentId--, content.text as postText
FROM post
JOIN content ON content.id = post.contentId
JOIN mb_user ON mb_user.id = content.creatorId
JOIN band_follower ON band_follower.bandId = post.bandId
JOIN band ON band.id = band_follower.bandId
LEFT JOIN comment ON comment.postId = post.id
LEFT JOIN content as commentContent ON commentContent.id = comment.contentId
LEFT JOIN mb_user as commentUser ON commentContent.creatorId = commentUser.id ) as all_posts
```

WHERE all\_posts.follower = 12  
AND all\_posts.private = false  
ORDER BY all\_posts.date DESC;

### 1.3 M03 Profile Posts

|               |                                                 |
|---------------|-------------------------------------------------|
| SQL Reference | Get posts from a user to build his profile page |
| -----         | -----                                           |
| Web Resource  | Not yet implemented                             |

sql

```
select * from post
join content
on post.contentId = content.id
join mb_user
on mb_user.id = $userId and mb_user.id = content.creatorId
left join comment
on comment.postId = post.id
join content a
on a.id = comment.contentId
join mb_user b
on b.id = a.creatorId;
```

### 1.4 M04 Band Profile Posts

|               |                                                 |
|---------------|-------------------------------------------------|
| SQL Reference | Get posts from a band to build its profile page |
| -----         | -----                                           |
| Web Resource  | R402                                            |

```

sql
select * from post
join content
on post.contentId = content.id
join band
on band.id = post.bandId and band.id = $bandId
left join comment
on comment.postId = post.id
join content a
on a.id = comment.contentId
join mb_user b
on b.id = a.creatorId;

```

### 1.5 M05 New User Post

|               |                         |  |
|---------------|-------------------------|--|
| SQL Reference | User creates a new post |  |
| -----         | -----                   |  |
| Web Resource  | R305                    |  |

[Transaction T01](#T01)

### 1.5 M05 New Band Post

|               |                                               |  |
|---------------|-----------------------------------------------|--|
| SQL Reference | User creates a new post in behalf of the band |  |
| -----         | -----                                         |  |
| Web Resource  | R405                                          |  |

[Transaction T02](#T02)

### 1.6 M06 New Comment

|               |                        |  |
|---------------|------------------------|--|
| SQL Reference | Send a message to User |  |
| -----         | -----                  |  |
| Web Resource  | R311                   |  |

[Transaction T02](#T02)

### 1.7 M07 Friend Chat

|               |                 |  |
|---------------|-----------------|--|
| SQL Reference | Search bar page |  |
| -----         | -----           |  |
| Web Resource  | R502            |  |

```

sql
SELECT message.id, creatorId, receiverId, text, date, isActive
FROM message
JOIN content ON content.id = message.contentId

```

WHERE (creatorId = \$userId AND receiverId = \$friendId) OR (creatorId = \$friendId  
AND receiverId = \$userId)  
ORDER BY date ASC;

### 1.8 M08 Band Chat

|               |                 |  |
|---------------|-----------------|--|
| SQL Reference | Search bar page |  |
| -----         | -----           |  |
| Web Resource  | R505            |  |

sql  
SELECT message.id, creatorId, text, date, isActive  
FROM message  
JOIN content ON content.id = message.contentId  
WHERE bandId = 2  
ORDER BY date ASC;

### 1.9 M09 Send Message to User

|               |                        |  |
|---------------|------------------------|--|
| SQL Reference | Send a message to User |  |
| -----         | -----                  |  |
| Web Resource  | R504                   |  |

[Transaction T03](#T03)

### 1.10 M10 Send Message to Band Chat

|               |                              |  |
|---------------|------------------------------|--|
| SQL Reference | Send a message the Band Chat |  |
| -----         | -----                        |  |
| Web Resource  | R507                         |  |

[Transaction T04](#T04)

### 1.11 M11 Search

|               |                 |  |
|---------------|-----------------|--|
| SQL Reference | Search bar page |  |
| -----         | -----           |  |
| Web Resource  |                 |  |

sql  
SELECT mb\_user.id, mb\_user.name as name, city.name as city, country.name as  
country, user\_follower.isActive as isFollowing  
FROM mb\_user  
LEFT JOIN city ON city.id = mb\_user.location  
LEFT JOIN country ON city.countryId = country.id

```

LEFT JOIN user_follower
ON user_follower.followedUserId = mb_user.id AND user_follower.followingUserId =
$userId
WHERE to_tsvector('simple', mb_user.name) @@ to_tsquery('simple', $text.':*')
ORDER BY isFollowing ASC;

```

### 1.12 M12 New Band

|               |                 |  |
|---------------|-----------------|--|
| SQL Reference | Create new band |  |
| -----         | -----           |  |
| Web Resource  | R414            |  |

[Transaction T04](#T04)

### 1.13 M13 Users List

|               |                  |  |
|---------------|------------------|--|
| SQL Reference | Get List of User |  |
| -----         | -----            |  |
| Web Resource  | R201             |  |

sql

```

SELECT *
FROM mb_user
ORDER BY admin desc,name
LIMIT 10
OFFSET $offset

```

### 1.14 M14 Reported Users

|               |                                                     |  |
|---------------|-----------------------------------------------------|--|
| SQL Reference | List of user reports, group by reports and warnings |  |
| -----         | -----                                               |  |
| Web Resource  | R203                                                |  |

sql

```

SELECT reports.user_id as user_id, reports.name as name, reports.sum as
number_of_reports, warnings.total as number_of_warnings

```

FROM

```

(SELECT user_id, name,sum(total) FROM

```

```

(-- user_reports
SELECT mb_user.id as user_id, mb_user.name as name, count(*) as total --times_r
FROM report
JOIN mb_user ON mb_user.id = report.reportedUserId
WHERE report.reportType <> 'band_report'
GROUP BY mb_user.id

-- user_reported_content
UNION ALL
SELECT mb_user.id as user_id, mb_user.name as name, count(*) as total --time
FROM report
JOIN content ON content.id = report.reportedContentId
JOIN mb_user ON content.creatorId = mb_user.id
GROUP BY mb_user.id) as total_reported

GROUP BY total_reported.user_id, total_reported.name) as reports

```

## LEFT JOIN

```

(SELECT mb_user.id as user_id, mb_user.name as name, count(*) as total --
times_warned
FROM warning
JOIN mb_user ON mb_user.id = warning.userId
GROUP BY mb_user.id) AS warnings on warnings.user_id = reports.user_id

```

ORDER BY number\_of\_reports DESC, number\_of\_warnings DESC;

### 1.15 M15 Reported Bands

|               |                                                     |
|---------------|-----------------------------------------------------|
| SQL Reference | List of band reports, group by reports and warnings |
| -----         | -----                                               |
| Web Resource  | R206                                                |

sql

```

SELECT reports.band_id as band_id, reports.name as band_name, reports.sum as
number_of_reports, warnings.total as number_of_warnings
FROM

```

```

(SELECT band_id, name, sum(total) FROM

```

```

(-- band_reports
SELECT band.id as band_id, band.name as name, count(*) as total --times_reported
FROM report
JOIN band ON band.id = report.reportedBandId
GROUP BY band.id

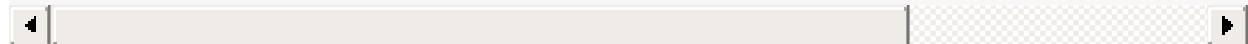
```

UNION ALL

-- band\_reported\_content

```
SELECT band.id as band_id, band.name as name, count(*) as total --times_published
FROM report
JOIN content ON content.id = report.reportedContentId
LEFT JOIN post ON post.contentId = content.id --AND post.bandId IS NOT NULL
JOIN band ON post.bandId = band.id
GROUP BY band.id) as total_reported
```

GROUP BY total\_reported.band\_id, total\_reported.name) as reports



LEFT JOIN

```
(SELECT band.id as band_id, band.name as name, count(*) as total --times_warned
FROM warning
```

```
JOIN band ON band.id = warning.bandId
```

```
GROUP BY band.id) AS warnings on warnings.band_id = reports.band_id
```

ORDER BY number\_of\_reports DESC, number\_of\_warnings DESC;

### 1.16 M16 User Reports

|               |                                        |  |
|---------------|----------------------------------------|--|
| SQL Reference | List of the reports of a specific user |  |
| -----         | -----                                  |  |
| Web Resource  | Not yet implemented                    |  |

sql

SELECT \*

FROM

```
(SELECT mb_user.id as user_id, mb_user.name as reportedUser, report.text as text,
users2.name as reporterUser, 'na' as contentText, 0 as postId, 0 as messageId, 0 as
commentId
```

```
FROM report
```

```
JOIN mb_user ON report.reportedUserId = mb_user.id
```

```
JOIN mb_user as users2 ON users2.id = report.reporterUserId
```

UNION ALL

```
SELECT mb_user.id as user_id,mb_user.name as reportedUser, report.text as text, user
FROM report
```

```

JOIN content ON content.id = report.reportedContentId
LEFT JOIN post ON post.contentId = content.id
LEFT JOIN message ON message.contentId = content.id
LEFT JOIN comment ON comment.contentId = content.id
JOIN mb_user ON content.creatorId = mb_user.id
JOIN mb_user as users2 ON users2.id = report.reporterUserId) as reports

```

WHERE reports.id = \$user\_id  
ORDER BY contentText;

### 1.17 M17 Band Reports

|               |                                        |
|---------------|----------------------------------------|
| SQL Reference | List of the reports of a specific band |
| -----         | -----                                  |
| Web Resource  | Not yet implemented                    |

sql

```

SELECT *
FROM
(SELECT band.id as band_id, band.name as reportedBand, report.text as complaint,
users2.name as reporterUser, 'na' as contentText, 0 as postId
FROM report
JOIN band ON report.reportedBandId = band.id
JOIN mb_user as users2 ON users2.id = report.reporterUserId

```

UNION ALL

```

SELECT band.id as band_id, band.name as reportedBand, report.text as complaint, users2.name as reporterUser, 'na' as contentText, 0 as postId
FROM report
JOIN content ON content.id = report.reportedContentId
LEFT JOIN post ON post.contentId = content.id
LEFT JOIN message ON message.contentId = content.id
JOIN band ON post.bandId = band.id
JOIN mb_user as users2 ON users2.id = report.reporterUserId) as reports

```

WHERE reports.band\_id = \$band\_id  
ORDER BY contentText;

## 2. Transactions



```

> Transactions needed to assure the integrity of the data.
### 2.1. Create consistency

### T01
T01	New User Post
Isolation level	REPEATABLE READ
Justification	In a new post, it's needed to add the data of the new post into

```

sql

```

BEGIN TRANSACTION;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

```

```
-- Insert content
```

```

INSERT INTO content (text, creatorId)
VALUES ($text, $creatorId);

```

```
-- Insert post
```

```

INSERT INTO post (private, contentId)
VALUES ($private, currval('content_id_seq'));

```

```

COMMIT;
...

```

## T02

| T02             | New Band Post                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Isolation level | REPEATABLE READ                                                                                                                                                                                                                                                                                                                                                                                                       |
| Justification   | In a new post, it's needed to add the data of the new post into <i>post</i> and <i>content</i> tables, in a single transaction in order to keep the consistency. The isolation level is Repeatable Read, because, otherwise, an update of <i>content_id_seq</i> could happen, due to an insert in the table <i>content</i> committed by a concurrent transaction, and as a result, inconsistent data would be stored. |

```

BEGIN TRANSACTION;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

```

```
-- Insert content
INSERT INTO content (text, creatorId)
VALUES ($text, $creatorId);

-- Insert post
INSERT INTO post (private, contentId, bandId)
VALUES ($private, currval('content_id_seq'), $bandId);

COMMIT;
```

## T03

| T03             | New Message to User                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Isolation level | REPEATABLE READ                                                                                                                                                                                                                                                                                                                                                                                                         |
| Justification   | In a new message, it's needed to add the data of the new message into <i>message</i> and <i>content</i> tables, in a single transaction in order to keep the consistency. The isolation level is Repeatable Read, because, otherwise, an update of content_id_seq could happen, due to an insert in the table <i>content</i> committed by a concurrent transaction, and as a result, inconsistent data would be stored. |

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

-- Insert content
INSERT INTO content (text, creatorId)
VALUES ($text, $creatorId);

-- Insert message
INSERT INTO message (contentId, receiverId)
VALUES (currval('content_id_seq'), $receiverId);

COMMIT;
```

## T04

| T04       | New Message to Band Chat |
|-----------|--------------------------|
| Isolation |                          |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| level         | REPEATABLE READ                                                                                                                                                                                                                                                                                                                                                                                                                |
| Justification | In a new message, it's needed to add the data of the new message into <i>message</i> and <i>content</i> tables, in a single transaction in order to keep the consistency. The isolation level is Repeatable Read, because, otherwise, an update of <i>content_id_seq</i> could happen, due to an insert in the table <i>content</i> committed by a concurrent transaction, and as a result, inconsistent data would be stored. |

```
BEGIN TRANSACTION;  
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ  
  
-- Insert content  
INSERT INTO content (text, creatorId)  
VALUES ($text, $creatorId);  
  
-- Insert message  
INSERT INTO message (contentId, bandId)  
VALUES (currval('content_id_seq'), $bandId);  
  
COMMIT;
```

# T05

| T05             | New Comment                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Isolation level | REPEATABLE READ                                                                                                                                                                                                                                                                                                                                                                                                                |
| Justification   | In a new comment, it's needed to add the data of the new comment into <i>comment</i> and <i>content</i> tables, in a single transaction in order to keep the consistency. The isolation level is Repeatable Read, because, otherwise, an update of <i>content_id_seq</i> could happen, due to an insert in the table <i>content</i> committed by a concurrent transaction, and as a result, inconsistent data would be stored. |

```
BEGIN TRANSACTION;  
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ  
  
-- Insert content  
INSERT INTO content (text, creatorId)
```

```
VALUES ($text, $creatorId);

-- Insert comment
INSERT INTO comment (contentId, postId)
VALUES (currval('content_id_seq'), $postId);

COMMIT;
```

## T06

| T06             | New Band                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Isolation level | REPEATABLE READ                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Justification   | In the process of creating a band, a <i>band_membership</i> must be created between the logged user and the band just created. To keep consistency these two inserts must be atomic. The isolation level is Repeatable Read, because, otherwise, an update of <i>band_id_seq</i> could happen, due to an insert in the table <i>band</i> committed by a concurrent transaction, and as a result, inconsistent data would be stored. |

```
```sql
```

```
BEGIN TRANSACTION;
```

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
```

```
INSERT INTO band(name) VALUES($band_name);
```

```
INSERT INTO band_membership (bandId, userId, isOwner)
```

```
VALUES (currval('band_id_seq'), $userId, true);
```

```
COMMIT;
```

```
### 2.1. Dependent Selects
```

```
### T07
```

```
| T07 | Get last 5 messages and unread messages count |
```

```
| ----- | ----- |
```

```
| Isolation level | SERIALIZABLE READ ONLY |
```

```
| Justification | In the middle of the transaction, the insertion of new rows in
```



---

```

sql
BEGIN TRANSACTION;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ ONLY

-- Get number of unread message notifications

SELECT count(*)
FROM user_notification
JOIN notification_trigger ON user_notification.notificationTriggerId =
notification_trigger.id AND notification_trigger.type = 'message'
WHERE visualizedDate IS NOT NULL
AND userId = $userId;

-- Get the 5 most recent message notifications received

WITH messageNotifs AS (
SELECT user_notification.notificationTriggerId, user_notification.text,
content.creatorId, notification_trigger.date, user_notification.visualizedDate
FROM user_notification
JOIN notification_trigger ON user_notification.notificationTriggerId =
notification_trigger.id AND notification_trigger.type = 'message'
JOIN message ON message.id = notification_trigger.originMessage
JOIN content ON content.id = message.contentId
WHERE user_notification.userId = $userId
)
SELECT *
FROM messageNotifs
WHERE (messageNotifs.date, messageNotifs.creatorId) IN (
SELECT MAX(messageNotifs.date), messageNotifs.creatorId
FROM messageNotifs
GROUP BY messageNotifs.creatorId
)
ORDER BY messageNotifs.date DESC
LIMIT 5;

COMMIT;
...
```

## T08

T08	
-----	--

	<b>Get last 8 notifications and unread notifications count</b>
Isolation level	SERIALIZABLE READ ONLY
Justification	In the middle of the transaction, the insertion of new rows in the user_notification table can occur, which implies that the information retrieved in both selects is different, consequently resulting in a Phantom Read. It's READ ONLY because it only uses Selects.

```

BEGIN TRANSACTION;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ ONLY

-- Get number of unread notifications

SELECT count(*)
FROM user_notification
JOIN notification_trigger ON user_notification.notificationTriggerId = notification_trigger.notificationTriggerId
WHERE visualizedDate IS NOT NULL
AND userId = $userId;

-- Get the 8 most recent notifications received

SELECT notification_trigger.id, user_notification.text, notification_trigger.date, notification_trigger.notificationTriggerId
FROM user_notification
JOIN notification_trigger ON user_notification.notificationTriggerId = notification_trigger.notificationTriggerId
WHERE user_notification.userId = $userId
ORDER BY notification_trigger.date DESC
LIMIT 8;

COMMIT;

```

GROUP1712, 15/04/2018

João Pinheiro, [up201104913@fe.up.pt](mailto:up201104913@fe.up.pt)

Leonardo Teixeira, [up201502848@fe.up.pt](mailto:up201502848@fe.up.pt)

Danny Soares, [up201505509@fe.up.pt](mailto:up201505509@fe.up.pt)

João Azevedo, [up201503256@fe.up.pt](mailto:up201503256@fe.up.pt)