A7: Integrity constraints. Indexes, triggers and user functions

1 Physical Schema

1.1 Análise da carga

Relação	Número tuplos	Observações	Crescimento esperado
Categoria	56 ~ 100	Espera-se que existam no mínimo cinquenta e seis categorias por forma a permitir que a uma instituição estejam associadas, no mínimo, quatro categorias; no máximo prevê-se a existência de algumas dezenas de categorias.	Não se espera qualquer alteração significativa ao número das categorias.
Instituicao	14	Espera-se um número de tuplos fixo, correspondente ao número de instituições de ensino da Universidade do Porto.	Não se prevê qualquer alteração no número das instituições da Universidade do Porto.
CategoriaInstituicao	56 ~ 1000	Espera-se que existam em média quatro ou cinco categorias associadas a cada instituição.	Não se prevê qualquer incremento significativo deste número, a não ser que mais categorias sejam adicionadas ao sistema por necessidade ou associadas às instituições existentes.
Utilizador	8 ~ 100	Prevê-se a existência, no mínimo, de oito utilizadores (cerca de metade do número de instituições) provenientes das várias instituições da Universidade do Porto; no máximo, prevê-se a existência de algumas dezenas de utilizadores.	prevê-se que este número sofra um crescimento logarítmico ao longo do tempo, mais acentuado nos primeiros meses de atividade do *KnowUP* e menos acentuado no futuro.
Administrador	1	Prevê-se a existência de um único utilizador com permissões administrativas.	Não se prevê qualquer alteração do número de administradores.
Moderador	2 ~ 10	De forma a moderar eficazmente as intervenções de todos os utilizadores, espera-se a existência de um moderador para cada cinquenta utilizadores, sendo necessárias algumas dezenas de moderadores. No caso do número de utilizadores registados ser mínimo, dois moderadores serão suficientes para moderar toda a comunidade.	Prevê-se que os moderadores tenham um crescimento proporcional ao aumento do número de utilizadores registados.

Relação	Número tuplos	Observações	Crescimento esperado
Pergunta	10 ~ 1000	Espera-se que existam pelo menos dez perguntas, publicadas pelo número mínimo de utilizadores previsto; no máximo, prevê-se a publicação de algumas centenas de perguntas, considerando que cada utilizador venha a publicar em média cinco perguntas.	Espera-se que as perguntas tenham um crescimento proporcional ao aumento do número de utilizadores registados, dado que mais utilizadores submetem mais questões.
Contribuicao	19 ~ 1000	Espera-se que existam pelo menos dezanove contribuições, publicadas pelo número mínimo de utilizadores previsto; no máximo, prevê-se a existência de contribuições na ordem dos milhares, partindo do pressuposto que cada utilizador responda a mais do que uma pergunta ou que comente mais do que uma pergunta.	Espera-se que as contribuições tenham um crescimento proporcional ao aumento do número de perguntas, de utilizadores e de respostas.
Resposta	13 ~ 1000	Espera-se que existam pelo menos treze respostas, publicadas pelo número mínimo de utilizadores previsto; espera-se que em média todas as perguntas tenham pelo menos uma resposta; no máximo, as perguntas deverão conter cinco respostas.	Espera-se que as respostas tenham um crescimento proporcional ao aumento do número de perguntas e do número de utilizadores registados.
ComentarioPergunta	2 ~ 1000	Espera-se que cada utilizador comente em média cinco perguntas; no máximo, as perguntas deverão conter cinco comentários.	Espera-se que os comentários às perguntas tenham um crescimento proporcional ao aumento do número de perguntas e do número de utilizadores registados.
ComentarioResposta	4 ~ 1000	Espera-se que cada utilizador comente em média cinco respostas; no máximo, as respostas deverão conter cinco comentários.	Espera-se que os comentários às respostas tenham um crescimento proporcional ao aumento do número de respostas e do número de utilizadores registados.
Seguidor	20 ~ 1000	Espera-se que cada autor de uma pergunta ou resposta seja seguidor dessa pergunta; se todos estes autores forem distintos, estima-se que as perguntas tenham em média dez ou mais seguidores.	Espera-se que os seguidores tenham um crescimento proporcional ao aumento do número de perguntas e do número de utilizadores registados.
VotoPergunta	37 ~ 1000	Espera-se, em média, que um utilizador vote em cerca de uma dezena de perguntas.	Espera-se que os votos tenham um crescimento proporcional ao aumento do número de perguntas e de utilizadores registados.

Relação	Número tuplos	Observações	Crescimento esperado
VotoResposta	36 ~ 1000	Espera-se, em média, que um utilizador vote em cerca de uma dezena de respostas.	Espera-se que os votos tenham um crescimento proporcional ao aumento do número de respostas e de utilizadores registados.
Conversa	6 ~ 1000	Espera-se, no caso mínimo, que cada utilizador tenha pelo menos uma conversa com outro utilizador; espera-se que cada utilizador tenha em média três conversas.	Espera-se que as conversas tenham um crescimento proporcional ao aumento do número de utilizadores registados.
Mensagem	15 ~ 1000	Espera-se, no caso mínimo, que cada conversa tenha pelo menos duas mensagens, enviadas por utilizadores distintos; no máximo, espera-se que que cada utilizador registado envie no máximo cinco mensagens por conversa.	Espera-se que a troca de mensagens tenha um crescimento proporcional ao aumento do número de conversas criadas entre utilizadores.
Report	4 ~ 10	Espera-se que nenhum membro venha a ser reportado; caso aconteça, espera-se que esse número não ultrapasse as dezenas.	Espera-se que os utilizadores reportados tenham um crescimento proporcional ao aumento do número de utilizadores.

Tabela 1: número de tuplos esperados para cada relação

1.2 Principais queries

Operação	Frequência	Chaves externas
Listar perguntas melhor classificadas	* * * * *	Utilizador.idUtilizador ⇔ Pergunta.idAutor
Listar perguntas mais recentes	* * * * *	Utilizador.idUtilizador ⇔ Pergunta.idAutor
Listar perguntas não respondidas	* * *	Resposta.idPergunta ⇔ Pergunta.idPergunta Utilizador.idUtilizador ⇔ Pergunta.idAutor
Listar instituições	* * * *	n/a
Listar categorias associadas às instituições	* * * *	Instituicao.idInstituicao ⇔ CategoriaInstituicao.idInstituicao CategoriaInstituicao.idCategoria ⇔ Categoria.idCategoria
Listar instituições associadas às categorias	* * *	Categoria.idCategoria ⇔ CategoriaInstituicao.idCategoria CategoriaInstituicao.idInstituicao ⇔ Instituicao.idInstituicao
Listar perguntas por categoria	* * * *	Instituicao.idInstituicao ⇔ Utilizador.idInstituicao Resposta.idPergunta ⇔ Pergunta.idPergunta Utilizador.idUtilizador ⇔ Pergunta.idAutor
Listar perguntas por instituição	* * *	Instituicao.idInstituicao ⇔ Utilizador.idInstituicao Resposta.idPergunta ⇔ Pergunta.idPergunta Utilizador.idUtilizador ⇔ Pergunta.idAutor
Listar perguntas por autor	* * *	Pergunta.idAutor ⇔ Utilizador.idUtilizador Pergunta.idPergunta ⇔ VotoPergunta.idPergunta

Operação	Frequência	Chaves externas
Listar respostas por autor	* *	Contribuicao.idAutor ⇔ Utilizador.idUtilizador Resposta.idResposta ⇔ VotoResposta.idResposta Contribuicao.idContribuicao ⇔ Resposta.idResposta Pergunta.idPergunta ⇔ Resposta.idPergunta
Listar respostas a uma pergunta	* * * * *	Instituicao.idInstituicao ⇔ Utilizador.idInstituicao Utilizador.idUtilizador ⇔ Pergunta.idAutor
Listar comentários a uma pergunta	* * * *	Contribuicao.idContribuicao ⇔ ComentarioPergunta.idComentario Utilizador.idUtilizador ⇔ Contribuicao.idAutor
Listar comentários a uma resposta	* * * *	Contribuicao.idContribuicao ⇔ ComentarioResposta.idComentario Utilizador.idUtilizador ⇔ Contribuicao.idAutor
Pesquisar perguntas	* * * *	Resposta.idPergunta ⇔ Pergunta.idPergunta Utilizador.idUtilizador ⇔ Pergunta.idAutor
Pesquisar utilizadores	* * *	n/a

Tabela 2: frequência das principais *queries* ao sistema

Listar perguntas melhor classificadas

```
CREATE VIEW PerguntasMelhorClassificadas AS
SELECT Pergunta.idPergunta,
       Utilizador.idUtilizador,
       Utilizador.primeiroNome || ' ' || Utilizador.ultimoNome AS
nomeUtilizador,
       Pergunta.titulo,
       Pergunta.descricao,
       Pergunta.ativa,
       COALESCE(TabelaRespostas.count, ) AS numeroRespostas,
       COALESCE(SUM(CASE WHEN valor = 1 THEN 1 ELSE  END), ) AS
votosPositivos,
       COALESCE(SUM(CASE WHEN valor = -1 THEN 1 ELSE  END), ) AS
votosNegativos,
       COALESCE(SUM(valor), ) AS pontuacao,
       EXTRACT(EPOCH FROM Pergunta.dataHora) AS dataHora
    FROM Pergunta
    LEFT JOIN VotoPergunta USING(idPergunta)
    LEFT JOIN (SELECT idPergunta, COUNT(*)
        FROM Resposta
        GROUP BY idPergunta)
        AS TabelaRespostas
        USING (idPergunta)
    JOIN Utilizador ON Utilizador.idUtilizador = Pergunta.idAutor
    GROUP BY TabelaRespostas.count, Pergunta.idPergunta,
Utilizador.idUtilizador
    ORDER BY pontuacao DESC;
```

Listar perguntas mais recentes

```
CREATE VIEW PerguntasMaisRecentes AS
```

```
SELECT Pergunta.idPergunta,
       Utilizador.idUtilizador,
       Utilizador.primeiroNome || ' ' || Utilizador.ultimoNome AS
nomeUtilizador,
       Pergunta.titulo,
       Pergunta.descricao,
       Pergunta.ativa,
       COALESCE(TabelaRespostas.count, ) AS numeroRespostas,
       COALESCE(SUM(CASE WHEN valor = 1 THEN 1 ELSE  END), ) AS
votosPositivos,
       COALESCE(SUM(CASE WHEN valor = -1 THEN 1 ELSE  END), ) AS
votosNegativos,
       COALESCE(SUM(valor), ) AS pontuacao,
       EXTRACT(EPOCH FROM Pergunta.dataHora) AS dataHora
    FROM Pergunta
    LEFT JOIN VotoPergunta USING(idPergunta)
    LEFT JOIN (SELECT idPergunta, COUNT(*)
        FROM Resposta
        GROUP BY idPergunta)
        AS TabelaRespostas
        USING (idPergunta)
    JOIN Utilizador ON Utilizador.idUtilizador = Pergunta.idAutor
    GROUP BY TabelaRespostas.count, Pergunta.idPergunta,
Utilizador.idUtilizador
    ORDER BY Pergunta.dataHora DESC;
```

Listar perguntas não respondidas

```
SELECT Pergunta.idPergunta,
       Utilizador.idUtilizador,
       Utilizador.primeiroNome || ' ' || Utilizador.ultimoNome AS
nomeUtilizador,
       Pergunta.titulo,
       Pergunta.descricao,
       Pergunta.ativa,
       COALESCE(TabelaRespostas.count, ) AS numeroRespostas,
       COALESCE(SUM(CASE WHEN valor = 1 THEN 1 ELSE  END), ) AS
votosPositivos,
       COALESCE(SUM(CASE WHEN valor = -1 THEN 1 ELSE  END), ) AS
votosNegativos,
       COALESCE(SUM(valor), ) AS pontuacao,
       EXTRACT(EPOCH FROM Pergunta.dataHora) AS dataHora
    FROM Pergunta
    LEFT JOIN VotoPergunta USING(idPergunta)
    LEFT JOIN (SELECT idPergunta, COUNT(*)
        FROM Resposta
        GROUP BY idPergunta)
        AS TabelaRespostas
        USING (idPergunta)
    LEFT JOIN Resposta USING(idPergunta)
    JOIN Utilizador ON Utilizador.idUtilizador = Pergunta.idAutor
```

```
    WHERE Resposta.idPergunta IS NULL
    GROUP BY TabelaRespostas.count, Pergunta.idPergunta,
Utilizador.idUtilizador
    ORDER BY Pergunta.dataHora DESC;
```

Listar instituições

```
SELECT Instituicao.idInstituicao,
       Instituicao.nome,
       Instituicao.sigla,
       COALESCE(COUNT(DISTINCT Utilizador.idUtilizador), ) AS
numeroUtilizadores,
       COALESCE(COUNT(DISTINCT CategoriaInstituicao.idCategoria), ) AS
numeroCategorias,
       COALESCE(COUNT(DISTINCT Pergunta.idPergunta), ) AS numeroPerguntas
    FROM Instituicao
    LEFT JOIN Utilizador USING(idInstituicao)
    LEFT JOIN CategoriaInstituicao USING(idInstituicao)
    LEFT JOIN Pergunta USING (idCategoria)
    GROUP BY Instituicao.idInstituicao
    ORDER BY Instituicao.nome;
```

Listar categorias associadas às instituições

```
SELECT Categoria.idCategoria, Categoria.nome
    FROM Instituicao
    JOIN CategoriaInstituicao USING(idInstituicao)
    JOIN Categoria USING(idCategoria)
    WHERE idInstituicao = :idInstituicao
    ORDER BY Categoria.nome ASC;
```

Listar instituições associadas às categorias

```
SELECT Instituicao.idInstituicao, Instituicao.sigla
    FROM Categoria
    JOIN CategoriaInstituicao USING(idCategoria)
    JOIN Instituicao USING(idInstituicao)
    WHERE Categoria.idCategoria = :idCategoria
    ORDER BY Instituicao.sigla ASC;
```

Listar perguntas por categoria

```
SELECT Pergunta.idPergunta,
       Utilizador.idUtilizador,
       Utilizador.primeiroNome || ' ' || Utilizador.ultimoNome AS
nomeUtilizador,
       Utilizador.username,
       Pergunta.titulo,
       Pergunta.descricao,
       Pergunta.ativa,
```

```
       COALESCE(TabelaRespostas.count, ) AS numeroRespostas,
       COALESCE(SUM(CASE WHEN valor = 1 THEN 1 ELSE  END), ) AS
votosPositivos,
       COALESCE(SUM(CASE WHEN valor = -1 THEN 1 ELSE  END), ) AS
votosNegativos,
       COALESCE(SUM(valor), ) AS pontuacao,
       EXTRACT(EPOCH FROM Pergunta.dataHora) AS dataHora
    FROM Pergunta
    LEFT JOIN VotoPergunta USING(idPergunta)
    LEFT JOIN (SELECT idPergunta, COUNT(*)
        FROM Resposta
        GROUP BY idPergunta)
        AS TabelaRespostas
        USING (idPergunta)
    JOIN Utilizador ON Utilizador.idUtilizador = Pergunta.idAutor
    WHERE Pergunta.idCategoria = :idCategoria
    GROUP BY Pergunta.idPergunta, TabelaRespostas.count,
Utilizador.idUtilizador
    ORDER BY Pergunta.dataHora DESC;
```

Listar perguntas por instituição

```
SELECT Pergunta.idPergunta,
       Utilizador.idUtilizador,
       Utilizador.primeiroNome || ' ' || Utilizador.ultimoNome AS
nomeUtilizador,
       Utilizador.username,
       Pergunta.titulo,
       Pergunta.descricao,
       Pergunta.ativa,
       COALESCE(TabelaRespostas.count, ) AS numeroRespostas,
       COALESCE(SUM(CASE WHEN valor = 1 THEN 1 ELSE  END), ) AS
votosPositivos,
       COALESCE(SUM(CASE WHEN valor = -1 THEN 1 ELSE  END), ) AS
votosNegativos,
       COALESCE(SUM(valor), ) AS pontuacao,
       EXTRACT(EPOCH FROM Pergunta.dataHora) AS dataHora
    FROM CategoriaInstituicao
    JOIN Pergunta USING(idCategoria)
    JOIN Utilizador ON Utilizador.idUtilizador = Pergunta.idAutor
    LEFT JOIN VotoPergunta USING(idPergunta)
    LEFT JOIN (SELECT idPergunta, COUNT(*)
        FROM Resposta
        GROUP BY idPergunta)
        AS TabelaRespostas
        USING (idPergunta)
    WHERE CategoriaInstituicao.idInstituicao = 12
    GROUP BY Pergunta.idPergunta, TabelaRespostas.count,
Utilizador.idUtilizador
    ORDER BY Pergunta.dataHora DESC;
```

Listar perguntas por autor

```
SELECT Pergunta.idPergunta,
       Pergunta.titulo,
       Pergunta.descricao,
       Pergunta.ativa,
       COALESCE(SUM(CASE WHEN valor = 1 THEN 1 ELSE  END), ) AS
votosPositivos,
       COALESCE(SUM(CASE WHEN valor = -1 THEN 1 ELSE  END), ) AS
votosNegativos,
       COALESCE(SUM(valor), ) AS pontuacao,
       EXTRACT(EPOCH FROM Pergunta.dataHora) AS dataHora
    FROM Utilizador
    JOIN Pergunta ON Utilizador.idUtilizador = Pergunta.idAutor
    LEFT JOIN VotoPergunta USING(idPergunta)
    WHERE Utilizador.idUtilizador = :idUtilizador
    GROUP BY Pergunta.idPergunta;
```

Listar respostas por autor

```
SELECT Resposta.idResposta,
       Pergunta.idPergunta,
       Pergunta.titulo,
       Contribuicao.descricao,
       Resposta.melhorResposta,
       Pergunta.ativa,
       COALESCE(SUM(CASE WHEN valor = 1 THEN 1 ELSE  END), ) AS
votosPositivos,
       COALESCE(SUM(CASE WHEN valor = -1 THEN 1 ELSE  END), ) AS
votosNegativos,
       COALESCE(SUM(valor), ) AS pontuacao,
       EXTRACT(EPOCH FROM Contribuicao.dataHora) AS dataHora
    FROM Contribuicao
    JOIN Resposta ON Resposta.idResposta = Contribuicao.idContribuicao
    JOIN Pergunta ON Pergunta.idPergunta = Resposta.idPergunta
    JOIN Utilizador ON Utilizador.idUtilizador = Contribuicao.idAutor
    LEFT JOIN VotoResposta USING(idResposta)
    WHERE Contribuicao.idAutor = :idUtilizador
    GROUP BY Contribuicao.idContribuicao, Pergunta.idPergunta,
Resposta.idResposta
    ORDER BY Resposta.idResposta DESC;
```

Listar respostas a determinada pergunta

```
SELECT Resposta.idResposta,
       Utilizador.idUtilizador,
       Utilizador.primeiroNome || ' ' || Utilizador.ultimoNome AS
nomeUtilizador,
       Utilizador.username,
       Instituicao.sigla,
       Contribuicao.descricao,
```

```
       COALESCE(TabelaComentarios.count, ) AS numeroComentarios,
       COALESCE(TabelaVotos.votosPositivos, ) AS votosPositivos,
       COALESCE(TabelaVotos.votosNegativos, ) AS votosNegativos,
       COALESCE(votosPositivos - votosNegativos, ) AS pontuacao,
       EXTRACT(EPOCH FROM Contribuicao.dataHora) AS dataHora
       Resposta.melhorResposta
    FROM Resposta
    JOIN Contribuicao ON Contribuicao.idContribuicao = Resposta.idResposta
    JOIN Utilizador ON Utilizador.idUtilizador = Contribuicao.idAutor
    LEFT JOIN (SELECT idResposta,
        SUM(CASE WHEN valor = 1 THEN 1 ELSE  END) AS votosPositivos,
        SUM(CASE WHEN valor = -1 THEN 1 ELSE  END) AS votosNegativos
        FROM VotoResposta
        GROUP BY idResposta)
        AS TabelaVotos
        USING (idResposta)
    LEFT JOIN (SELECT idResposta, COUNT(*)
        FROM ComentarioResposta
        GROUP BY idResposta)
        AS TabelaComentarios
        USING (idResposta)
    WHERE Resposta.idPergunta = :idPergunta
    ORDER BY Contribuicao.dataHora DESC;
```

Listar comentários a uma pergunta

```
SELECT ComentarioPergunta.idComentario,
       Utilizador.idUtilizador,
       Utilizador.primeiroNome || ' ' || Utilizador.ultimoNome AS
nomeUtilizador,
       Contribuicao.descricao,
       EXTRACT(EPOCH FROM Contribuicao.dataHora) AS dataHora
    FROM ComentarioPergunta
    JOIN Contribuicao ON Contribuicao.idContribuicao =
ComentarioPergunta.idComentario
    JOIN Utilizador ON Utilizador.idUtilizador = Contribuicao.idAutor
    WHERE ComentarioPergunta.idPergunta = :idPergunta;
```

Listar comentários a uma resposta

```
SELECT ComentarioResposta.idComentario,
       Utilizador.idUtilizador,
       Utilizador.primeiroNome || ' ' || Utilizador.ultimoNome AS
nomeUtilizador,
       Contribuicao.descricao,
       EXTRACT(EPOCH FROM Contribuicao.dataHora) AS dataHora
    FROM ComentarioResposta
    JOIN Contribuicao ON Contribuicao.idContribuicao =
ComentarioResposta.idComentario
    JOIN Utilizador ON Utilizador.idUtilizador = Contribuicao.idAutor
```

```
    WHERE ComentarioResposta.idResposta = :idResposta;
```

Pesquisar perguntas

```
SELECT QueryPrincipal.idPergunta,
       QueryPrincipal.idUtilizador,
       QueryPrincipal.nomeUtilizador,
       QueryPrincipal.titulo,
       QueryPrincipal.dataHora,
       QueryPrincipal.ativa,
       ts_headline('portuguese', QueryPrincipal.conteudo, query) AS
descricao,
       QueryPrincipal.rank
    FROM (SELECT PerguntasPesquisa.*, query, ts_rank_cd(pesquisa, query) AS
rank
          FROM PerguntasPesquisa, plainto_tsquery('portuguese',
:stringPesquisa) AS query
          WHERE query @@ pesquisa
          ORDER BY rank DESC) AS QueryPrincipal;
```

Pesquisar utilizadores

```
SELECT UtilizadoresPesquisa.idUtilizador,
       UtilizadoresPesquisa.username,
       UtilizadoresPesquisa.nomeUtilizador,
       UtilizadoresPesquisa.email,
       ts_rank_cd(UtilizadoresPesquisa.pesquisa, query) AS rank
    FROM UtilizadoresPesquisa, plainto_tsquery('english', :stringPesquisa)
AS query
    WHERE query @@ pesquisa
    ORDER BY rank DESC;
```

1.3 Principais operações

Operação	Frequência	Tipo	Relação
Editar informações do utilizador	* *	UPDATE	Utilizador
Editar conteúdo da pergunta	* *	UPDATE	Pergunta
Editar conteúdo da resposta	* * *	UPDATE	Resposta
Escolher melhor resposta	* * * *	UPDATE	Resposta
Fechar pergunta	* * *	UPDATE	Pergunta
Seguir pergunta	* * * * *	INSERT	Seguidor
Unfollow pergunta	* * * *	INSERT	Seguidor
Apagar utilizador	* *	UPDATE	Utilizador
Banir utilizador	* *	UPDATE	Utilizador
Denunciar utilizador	* *	INSERT	Report

Tabela 3: frequência das principais operações

Editar informações do utilizador

```
UPDATE Utilizador
SET primeiroNome = COALESCE(:primeiroNome, primeiroNome),
    ultimoNome =  COALESCE(:novoUltimoNome, ultimoNome),
    email =  COALESCE(:novoEmail, email),
    localidade =  COALESCE(:novaLocalidade, localidade),
    codigoPais =  COALESCE(:novoCodigoPais, codigoPais)
WHERE idUtilizador = :idUtilizador;
```

Editar conteúdo da pergunta

```
UPDATE Pergunta
SET titulo = COALESCE(:novoTitulo, titulo),
    descricao = COALESCE(:novaDescricao, descricao)
WHERE idPergunta = :idPergunta;
```

Editar conteúdo da resposta

```
UPDATE Contribuicao
SET descricao = COALESCE(:novaDescricao, descricao)
WHERE idContribuicao = :idResposta;
```

Escolher melhor resposta

```
UPDATE Resposta
SET melhorResposta = TRUE
WHERE idResposta = :idResposta;
```

Fechar pergunta

```
UPDATE Pergunta
SET ativa = FALSE
WHERE idPergunta = :idPergunta;
```

Seguir pergunta

```
INSERT INTO Seguidor(idPergunta, idAutor)
VALUES(:idPergunta, :idAutor);
```

Unfollow pergunta

```
DELETE FROM Seguidor
WHERE idSeguidor = :idSeguidor
AND idPergunta = :idPergunta;
```

Apagar utilizador

```
UPDATE Utilizador
SET removido = TRUE
WHERE idUtilizador = :idUtilizador;
```

Banir utilizador

```
UPDATE Utilizador
SET ativo = FALSE
WHERE idUtilizador = :idUtilizador;
```

Denunciar utilizador

```
INSERT INTO Report(idModerador, idUtilizador, descricao)
VALUES(:idModerador, :idUtilizador, :descricao);
```

1.4 Índices

Categoria_IDX_Pesquisa

O índice **Categoria_IDX_Pesquisa** facilita operações de pesquisa sobre as categorias de perguntas existentes. Os resultados serão ordenados por ordem alfabética. Considerou-se um índice do tipo **hash**, já que esta operação se baseia num critério de igualdade.

```
CREATE INDEX Categoria_IDX_Pesquisa ON Categoria USING hash(nome);
```

Contribuicao_IDX_MaisRecentes

O índice **Contribuicao_IDX_MaisRecentes** facilita a listagem e ordenação das contribuições mais recentes. Considerou-se um índice do tipo **btree** já que esta operação se baseia na comparação de grandeza entre as datas de publicação das contribuições e a data corrente.

```
CREATE INDEX Contribuicao_IDX_MaisRecentes ON Contribuicao USING
btree(dataHora);
```

Pergunta_IDX_Lookup

O índice **Pergunta_IDX_Lookup** traduz a relação **pergunta ⇔ autor**. Considerou-se um índice do tipo **btree** já que se trata de um índice composto entre uma chave primária e uma chave externa da mesma tabela.

```
CREATE INDEX Pergunta_IDX_Lookup ON Pergunta USING btree(idPergunta,
idAutor);
```

Pergunta_IDX_MaisRecentes

O índice **Pergunta_IDX_MaisRecentes** facilita a listagem e ordenação das perguntas mais recentes. Considerou-se um índice do tipo **btree** já que esta operação se baseia na comparação de grandeza entre as datas de publicação das perguntas e a data corrente.

```
CREATE INDEX Pergunta_IDX_MaisRecentes ON Pergunta USING btree(dataHora);
```

Pergunta_IDX_Pesquisa

O índice **Pergunta_IDX_Pesquisa** facilita operações de pesquisa *full-text* nos corpos, nos títulos e nas respostas das perguntas existentes no sistema, apresentando os resultados ordenados por relevância. Considerou-se um índice do tipo **gin** já que este tipo de pesquisa recorre a vectores de palavras e permite verificar se determinada palavra ou expressão está contida num dos três campos acima apresentados.

```
CREATE INDEX Pergunta_IDX_Pesquisa ON PerguntasPesquisa USING gin(pesquisa);
```

Resposta_IDX_Lookup

O índice **Resposta_IDX_Lookup** traduz a relação **respostas ⇔ pergunta**. Considerou-se um índice do tipo **btree** já que se trata de um índice composto entre uma chave primária e uma chave externa da mesma tabela.

```
CREATE INDEX Resposta_IDX_Lookup ON Resposta USING btree(idResposta,
idPergunta);
```

Utilizador_IDX_Pesquisa

O índice **Utilizador_IDX_Pesquisa** facilita operações de pesquisa *full-text* sobre os utilizadores (nos campos *username*, endereço de *e-mail*, nome completo), apresentando os resultados ordenados por relevância. Considerou-se um índice do tipo **gin**, já que este tipo de pesquisa recorre a vectores de palavras para verificar se os termos de pesquisa introduzidos pelo utilizador estão contidos numa dos quatro campos apresentados em cima.

```
CREATE INDEX Utilizador_IDX_Pesquisa ON UtilizadoresPesquisa USING
gin(pesquisa);
```

Utilizador_IDX_Username

O índice **Utilizador_IDX_Username** facilita a autenticação dos utilizadores através da combinação *username*/palavra-passe. Considerou-se um índice do tipo **hash** já que esta operação se baseia num critério de igualdade.

```
CREATE INDEX Utilizador_IDX_Username ON Utilizador USING hash(username);
```

1.5 Vistas materializadas

PerguntasPesquisa

Esta vista materializada permite a agregação de vários elementos do documento “Pergunta” (título, descrição, respostas) numa única coluna para realizar pesquisas *full-text* sobre as perguntas existentes no sistema.

```
CREATE MATERIALIZED VIEW PerguntasPesquisa AS
SELECT QueryPrincipal.idPergunta,
       Utilizador.idUtilizador,
       Utilizador.primeiroNome || ' ' || Utilizador.ultimoNome AS
nomeUtilizador,
       QueryPrincipal.titulo,
       QueryPrincipal.conteudo,
       QueryPrincipal.numeroRespostas,
       QueryPrincipal.dataHora,
       QueryPrincipal.ativa,
       to_tsvector('portuguese', conteudo) AS pesquisa
    FROM (SELECT Pergunta.idPergunta,
          Pergunta.titulo,
          Pergunta.idAutor,
          Pergunta.ativa,
          Pergunta.titulo || ' ' ||
          COALESCE(Pergunta.descricao, '') ||
          COALESCE(string_agg(Contribuicao.descricao, ' '), '') AS conteudo,
          COALESCE(COUNT (DISTINCT Resposta.idResposta), ) AS
numeroRespostas,
          EXTRACT(EPOCH FROM Pergunta.dataHora) AS dataHora
          FROM Pergunta
          LEFT JOIN Resposta USING(idPergunta)
          LEFT JOIN Contribuicao ON Contribuicao.idContribuicao =
Resposta.idResposta
          GROUP BY idPergunta) AS QueryPrincipal
    LEFT JOIN Utilizador ON Utilizador.idUtilizador =
QueryPrincipal.idAutor;
```

UtilizadoresPesquisa

Esta vista materializada permite a agregação de vários elementos da tabela “Utilizador” (*username*, endereço de *e-mail*, primeiro nome, último nome) numa única coluna para realizar pesquisas *full-text* sobre os utilizadores existentes no sistema.

```
CREATE MATERIALIZED VIEW UtilizadoresPesquisa AS
SELECT Utilizador.idUtilizador,
       Utilizador.username,
       Utilizador.primeiroNome || ' ' || Utilizador.ultimoNome AS
nomeUtilizador,
       Utilizador.email,
       to_tsvector('english', Utilizador.username || ' ' ||
           Utilizador.primeiroNome || ' ' ||
```

```
            Utilizador.ultimoNome || ' ' ||
            Utilizador.email) AS pesquisa
    FROM Utilizador;
```

2 Triggers

Restrição (regra de negócio)	Trigger	Relação
"Um utilizador não pode atribuir votos às suas perguntas"	TRIGGER_votarPropriaPergunta	VotoPergunta
"Um utilizador não pode atribuir votos às suas respostas"	TRIGGER_votarPropriaResposta	VotoResposta
"Um utilizador não pode responder às suas perguntas"	TRIGGER_responderPergunta	Resposta
"Um utilizador não pode responder a uma pergunta que tenha sido fechada pelo autor."	TRIGGER_responderPergunta	Resposta
"Deve existir no máximo uma melhor resposta por pergunta"	TRIGGER_melhorResposta	Resposta
"Um moderador não pode ser simultaneamente um administrador"	TRIGGER_overlapAdministrador	Administrador
"Um administrador não pode ser simultaneamente um moderador"	TRIGGER_overlapModerador	Moderador
"Um utilizador segue automaticamente as suas perguntas"	TRIGGER_autofollowpergunta	Pergunta
"Um utilizador que responda a uma pergunta segue automaticamente essa pergunta"	TRIGGER_autofollowResposta	Resposta
"Um utilizador que comente uma pergunta segue automaticamente essa pergunta"	TRIGGER_autofollowComentario	ComentarioPergunta
"A vista materializada PerguntasPesquisa deve ser atualizada sempre que for realizada uma operação sobre as tabelas *Pergunta* ou *Resposta*"	TRIGGER_atualizarPesquisa	Pergunta, Resposta
"A vista materializada UtilizadoresPesquisa deve ser atualizada sempre que for realizada uma operação sobre a tabela *Utilizador*"	TRIGGER_atualizarUtilizadores	Utilizador

2.1 TRIGGER_votarPropriaPergunta

Este *trigger* impede que os autores das perguntas insiram votos nas suas próprias perguntas.

```
CREATE OR REPLACE FUNCTION votarPropriaPergunta()
RETURNS TRIGGER AS $votarPropriaPergunta$
DECLARE
    AutorPergunta INTEGER;
BEGIN
    SELECT Pergunta.idAutor INTO AutorPergunta
    FROM Pergunta WHERE Pergunta.idPergunta = NEW.idPergunta;
    IF (AutorPergunta = NEW.idAutor) THEN
        RAISE EXCEPTION 'não pode classificar as suas próprias perguntas!';
        RETURN NULL;
```

```
    END IF;
    RETURN NEW;
END;

$votarPropriaPergunta$ LANGUAGE plpgsql;

CREATE TRIGGER TRIGGER_votarPropriaPergunta
    BEFORE INSERT ON VotoPergunta
    FOR EACH ROW
    EXECUTE PROCEDURE votarPropriaPergunta();
```

2.2 TRIGGER_votarPropriaResposta

Este *trigger* impede que os autores das respostas insiram votos nas suas próprias respostas.

```
CREATE OR REPLACE FUNCTION votarPropriaResposta()
RETURNS TRIGGER AS $votarPropriaResposta$
DECLARE
    AutorResposta INTEGER;
BEGIN
    SELECT Contribuicao.idAutor INTO AutorResposta
    FROM Resposta
    JOIN Contribuicao ON Contribuicao.idContribuicao = Resposta.idResposta
    WHERE Resposta.idResposta = NEW.idResposta;
    IF (AutorResposta = NEW.idAutor) THEN
        RAISE EXCEPTION 'não pode classificar as suas próprias respostas!';
        RETURN NULL;
    END IF;
    RETURN NEW;
END;

$votarPropriaResposta$ LANGUAGE plpgsql;

CREATE TRIGGER TRIGGER_votarPropriaResposta
    BEFORE INSERT ON VotoResposta
    FOR EACH ROW
    EXECUTE PROCEDURE votarPropriaResposta();
```

2.3 TRIGGER_responderPergunta

Este *trigger* impede que os autores das perguntas respondam às suas próprias perguntas. Por outro lado, impede também que sejam publicadas respostas nas perguntas fechadas pelos respectivos autores.

```
CREATE OR REPLACE FUNCTION responderPergunta()
RETURNS TRIGGER AS $responderPergunta$
DECLARE
    AutorPergunta INTEGER;
```

```
    AutorResposta INTEGER;
    PerguntaActiva BOOLEAN;
BEGIN
    SELECT Pergunta.ativa INTO PerguntaActiva
    FROM Pergunta WHERE NEW.idPergunta = Pergunta.idPergunta
    LIMIT 1;
    IF (NOT PerguntaActiva) THEN
        RAISE EXCEPTION 'não pode responder a uma pergunta fechada!';
        RETURN NULL;
    END IF;
    SELECT Pergunta.idAutor INTO AutorPergunta
    FROM Pergunta WHERE Pergunta.idPergunta = NEW.idPergunta
    LIMIT 1;
    SELECT Contribuicao.idAutor INTO AutorResposta
    FROM Contribuicao WHERE Contribuicao.idContribuicao = NEW.idResposta
    LIMIT 1;
    IF (AutorResposta = AutorPergunta) THEN
        RAISE EXCEPTION 'não pode responder às suas próprias perguntas!';
        RETURN NULL;
    END IF;
    RETURN NEW;
END;

$responderPergunta$ LANGUAGE plpgsql;

CREATE TRIGGER TRIGGER_responderPergunta
    BEFORE INSERT ON Resposta
    FOR EACH ROW
    EXECUTE PROCEDURE responderPergunta();
```

2.4 TRIGGER_melhorResposta

Este *trigger* garante que cada pergunta tem uma e uma só melhor resposta escolhida pelo seu autor. Sempre que for seleccionada uma nova melhor resposta a determinada pergunta, este *trigger* coloca a *flag* **melhorResposta** = *false* da resposta anterior e atualiza a *flag* da nova, colocando a *true*.<

```
CREATE OR REPLACE FUNCTION melhorResposta()
RETURNS TRIGGER AS $melhorResposta$
BEGIN
    IF EXISTS (SELECT 1 FROM Resposta WHERE Resposta.melhorResposta AND
Resposta.idPergunta = NEW.idPergunta) THEN
        UPDATE Resposta SET melhorResposta = FALSE
        WHERE Resposta.melhorResposta AND Resposta.idPergunta =
NEW.idPergunta;
    END IF;
    RETURN NEW;
END;

$melhorResposta$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER TRIGGER_melhorResposta
    BEFORE UPDATE OF melhorResposta ON Resposta
    FOR EACH ROW
    WHEN (NEW.melhorResposta AND NOT OLD.melhorResposta)
    EXECUTE PROCEDURE melhorResposta();
```

2.5 TRIGGER_overlapAdministrador

Este *trigger* impede que os moderadores sejam simultaneamente administradores. Um utilizador não pode ser promovido a administrador por meio de **INSERT** quando já se encontra na tabela **Moderador**; a respetiva linha da tabela deve ser apagada antes de ser realizada cqualquer operação.

```
CREATE OR REPLACE FUNCTION overlapAdministrador()
RETURNS TRIGGER AS $overlapAdministrador$
BEGIN
    IF EXISTS (SELECT 1 FROM Moderador WHERE Moderador.idModerador =
NEW.idAdministrador) THEN
        RETURN NULL;
    END IF;
    RETURN NEW;
END;

$overlapAdministrador$ LANGUAGE plpgsql;

CREATE TRIGGER TRIGGER_overlapAdministrador
    BEFORE INSERT OR UPDATE ON Administrador
    FOR EACH ROW
    EXECUTE PROCEDURE overlapAdministrador();
```

2.6 TRIGGER_overlapModerador

Este *trigger* impede que os administradores sejam simultaneamente moderadores. Um administrador não pode ser despromovido a moderador por meio de **INSERT** quando já se encontra na tabela **Administrador**; a respetiva linha da tabela deve ser apagada antes de ser realizada qualquer operação.

```
CREATE OR REPLACE FUNCTION overlapModerador()
RETURNS TRIGGER AS $overlapModerador$
BEGIN
    IF EXISTS (SELECT 1 FROM Administrador WHERE
Administrador.idAdministrador = NEW.idModerador) THEN
        RETURN NULL;
    END IF;
    RETURN NEW;
END;

$overlapModerador$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER TRIGGER_overlapModerador
    BEFORE INSERT OR UPDATE ON Moderador
    FOR EACH ROW
    EXECUTE PROCEDURE overlapModerador();
```

2.7 TRIGGER_autofollowPergunta

Este *trigger* adiciona automaticamente o autor de uma pergunta à tabela dos seguidores dessa pergunta.

```
CREATE OR REPLACE FUNCTION autofollowPergunta()
RETURNS TRIGGER AS $autofollowPergunta$
BEGIN
    INSERT INTO Seguidor(idSeguidor, idPergunta, dataInicio, dataAcesso)
    VALUES(NEW.idAutor, NEW.idPergunta, NEW.dataHora, NEW.dataHora);
    RETURN NEW;
    EXCEPTION WHEN unique_violation THEN
        RETURN NEW;
END;

$autofollowPergunta$ LANGUAGE plpgsql;

CREATE TRIGGER TRIGGER_autofollowPergunta
    AFTER INSERT ON Pergunta
    FOR EACH ROW
    EXECUTE PROCEDURE autofollowPergunta();
```

2.8 TRIGGER_autofollowResposta

Este *trigger* adiciona automaticamente o autor de uma resposta à tabela dos seguidores dessa pergunta.

```
CREATE OR REPLACE FUNCTION autofollowResposta()
RETURNS TRIGGER AS $autofollowResposta$
BEGIN
    INSERT INTO Seguidor
    SELECT Contribuicao.idAutor, NEW.idPergunta, Contribuicao.dataHora,
Contribuicao.dataHora
    FROM Contribuicao
    WHERE Contribuicao.idContribuicao = NEW.idResposta
    LIMIT 1;
    RETURN NEW;
    EXCEPTION WHEN unique_violation THEN
        RETURN NEW;
END;

$autofollowResposta$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER TRIGGER_autofollowResposta
    AFTER INSERT ON Resposta
    FOR EACH ROW
    EXECUTE PROCEDURE autofollowResposta();
```

2.9 TRIGGER_autofollowComentario

Este *trigger* adiciona automaticamente o autor de um comentário à tabela dos seguidores dessa pergunta.

```
CREATE OR REPLACE FUNCTION autofollowComentario()
RETURNS TRIGGER AS $autofollowComentario$
BEGIN
    INSERT INTO Seguidor
    SELECT Contribuicao.idAutor, NEW.idPergunta, Contribuicao.dataHora,
Contribuicao.dataHora
    FROM Contribuicao
    WHERE Contribuicao.idContribuicao = NEW.idPergunta
    LIMIT 1;
    RETURN NEW;
    EXCEPTION WHEN unique_violation THEN
        RETURN NEW;
END;

$autofollowComentario$ LANGUAGE plpgsql;

CREATE TRIGGER TRIGGER_autofollowComentario
    AFTER INSERT ON ComentarioPergunta
    FOR EACH ROW
    EXECUTE PROCEDURE autofollowComentario();
```

2.10 TRIGGER_atualizarPesquisa

Este *trigger* atualiza automaticamente a vista materializada **PerguntasPesquisa** sempre que for realizada uma operação sobre as tabelas *Pergunta* ou *Resposta*.

```
CREATE OR REPLACE FUNCTION atualizarPesquisa()
RETURNS TRIGGER AS $atualizarPesquisa$
BEGIN
    REFRESH MATERIALIZED VIEW PerguntasPesquisa;
    RETURN NULL;
END;

$atualizarPesquisa$ LANGUAGE plpgsql;

CREATE TRIGGER TRIGGER_atualizarPesquisaPergunta
    AFTER INSERT OR UPDATE OR DELETE OR TRUNCATE
    ON Pergunta
```

```
    FOR EACH STATEMENT
    EXECUTE PROCEDURE atualizarPesquisa();

CREATE TRIGGER TRIGGER_atualizarPesquisaResposta
    AFTER INSERT OR UPDATE OR DELETE OR TRUNCATE
    ON Resposta
    FOR EACH STATEMENT
    EXECUTE PROCEDURE atualizarPesquisa();
```

2.11 TRIGGER_atualizarUtilizadores

Este *trigger* atualiza automaticamente a vista materializada **UtilizadoresPesquisa** sempre que for realizada uma operação sobre a tabela *Utilizador*.

```
CREATE OR REPLACE FUNCTION atualizarUtilizadores()
RETURNS TRIGGER AS $atualizarUtilizadores$
BEGIN
    REFRESH MATERIALIZED VIEW UtilizadoresPesquisa;
    RETURN NULL;
END;
$atualizarUtilizadores$ LANGUAGE plpgsql;

CREATE TRIGGER TRIGGER_atualizarUtilizadores
    AFTER INSERT OR UPDATE OR DELETE OR TRUNCATE
    ON Utilizador
    FOR EACH STATEMENT
    EXECUTE PROCEDURE atualizarUtilizadores();
```

3 User-defined Functions

3.1 registarVotoPergunta(idPergunta, idAutor, valor)

Esta função permite inserir um voto em determinada pergunta ou alterar a classificação atribuída, caso este já exista; se a classificação for nula, não altera a tabela.

```
CREATE OR REPLACE FUNCTION registarVotoPergunta(INTEGER, INTEGER, INTEGER)
RETURNS BOOLEAN AS $registarVotoPergunta$
BEGIN
    IF ($3 = ) THEN
        RETURN FALSE;
    END IF;
    IF EXISTS(SELECT 1 FROM VotoPergunta WHERE idPergunta = $1 AND idAutor =
$2) THEN
        UPDATE VotoPergunta
        SET valor = $3 WHERE (idPergunta = $1 AND idAutor = $2);
        RETURN TRUE;
    ELSE
```

```
        INSERT INTO VotoPergunta(idPergunta, idAutor, valor)
        VALUES ($1, $2, $3);
        RETURN TRUE;
    END IF;
    RETURN FALSE;
END;

$registarVotoPergunta$ LANGUAGE plpgsql;
```

3.2 registarVotoResposta(idResposta, idAutor, valor)

Esta função permite inserir um voto em determinada resposta ou alterar a classificação atribuída, caso este já exista; se a classificação for nula, não altera a tabela.

```
CREATE OR REPLACE FUNCTION registarVotoResposta(INTEGER, INTEGER, INTEGER)
RETURNS BOOLEAN AS $registarVotoResposta$
BEGIN
    IF ($3 = ) THEN
        RETURN FALSE;
    END IF;
    IF EXISTS(SELECT 1 FROM VotoResposta WHERE idResposta = $1 AND idAutor =
$2) THEN
        UPDATE VotoResposta
        SET valor = $3 WHERE (idResposta = $1 AND idAutor = $2);
        RETURN TRUE;
    ELSE
        INSERT INTO VotoResposta(idResposta, idAutor, valor)
        VALUES ($1, $2, $3);
        RETURN TRUE;
    END IF;
    RETURN FALSE;
END;

$registarVotoResposta$ LANGUAGE plpgsql;
```

3.3 visitarPergunta(idPergunta, idUtilizador)

Esta função regista a visita de um utilizador a determinada pergunta, isto é, atualiza a data do último acesso do utilizador para a data actual, se este for seguidor da pergunta.

```
CREATE OR REPLACE FUNCTION visitarPergunta(INTEGER, INTEGER)
RETURNS VOID AS $visitarPergunta$
BEGIN
    UPDATE Seguidor
    SET dataAcesso = now()
    WHERE idPergunta = $1 AND idSeguidor = $2;
    UPDATE Pergunta
    SET visualizacoes = visualizacoes + 1
```

```
    WHERE idPergunta = $1;
    RETURN;
END;

$visitarPergunta$ LANGUAGE plpgsql;
```

3.4 lerMensagens(idConversa, idUtilizador)

Esta função regista a leitura das mensagens por parte de um dos intervenientes da conversa, isto é, atualiza a data do último acesso de um dos intervenientes.

```
CREATE OR REPLACE FUNCTION lerMensagens(INTEGER, INTEGER)
RETURNS VOID AS $lerMensagens$
BEGIN
    UPDATE Conversa
    SET ultimoAcesso1 = now()
    WHERE idConversa = $1 AND idUtilizador1 = $2;
    UPDATE Conversa
    SET ultimoAcesso2 = now()
    WHERE idConversa = $1 AND idUtilizador2 = $2;
    RETURN;
END;

$lerMensagens$ LANGUAGE plpgsql;
```

— Grupo 25

[< Back to KnowUP!]

From:
http://lbaw.fe.up.pt/201516/ - **L B A W :: WORK**

Permanent link:
http://lbaw.fe.up.pt/201516/doku.php/lbaw1525/proj/a7

Last update: **2016/04/19 11:20**