ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Licenciatura em Engenharia Informática

Licenciatura em Segurança Informática em Redes de Computadores

Base de Dados

Grupo 111

Fábio da Cunha (LSIRC) - 8210619

Hélder Henrique Sousa Dias Branco (LEI) - 8200302

Roger Seiji Hernandez Nakauchi (LSIRC) – 8210005

Junho de 2023

Resumo

O Trabalho Prático “Gestão local de uma App de Comunicação” é apresentado como resposta à proposta de trabalho da Unidade Curricular de Base de Dados lecionada na Escola Superior de Tecnologia e Gestão do Instituto Politécnico do Porto para o ano letivo de 2022/2023.

Ao longo do trabalho foram desenvolvidas soluções para suportar alguns serviços de uma app de comunicação. Procedeu-se à análise da aplicação, através do qual foram retiradas informações para a construção de um modelo de dados, para implementação em base de dados. Seguidamente foi criada a documentação de suporte e por último a implementação física em base de dados com recurso a SQL Server.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

No âmbito da Unidade Curricular de Base de Dados foi proposto o desenvolvimento de um sistema de gestão de base de dados para uma aplicação de comunicação que armazena os dados localmente no dispositivo.

Para a resolução do problema proposto será utilizada a metodologia abordada no livro Database Systems (A Pratical Approach to Design, Implementation, and Management - 6th edition), iniciando o processo pela elaboração de um modelo concetual, seguindo-se o modelo lógico e por fim o modelo físico.

1.2. Apresentação do Caso de Estudo

A empresa de software pretende implementar um sistema de gestão de base de dados de forma a suportar a gestão e armazenamento de dados da aplicação localmente num dispositivo.

Através da criação de uma base de dados, para suporte à informação necessária aos processos acima descritos, a empresa pretende que a aplicação consiga obter rapidamente elementos que lhe permita responder às seguintes questões:

- Qual o contacto para o qual são enviadas mais mensagens.
- Liste o número de mensagens recebidas por semana relativamente ao ano de 2022.
- Lista dos grupos de contactos, ordenada por ordem descendente da quantidade de participantes no grupo.
- Qual a hora do dia em que envia/recebe mais mensagens?
- Outras que achar relevantes.

1.3. Motivação e Objetivos

O objetivo fulcral deste trabalho, como já anteriormente referido, além de aplicar e demonstrar os conteúdos adquiridos, é de desenvolver a base de dados que satisfaça as necessidades de negócio pretendido. Um dos muitos benefícios e capacidades que esta unidade curricular e este projeto nos propõem é a capacidade de interpretar as necessidades de uma certa empresa ou projeto e conseguir de forma abstrata elaborar e implementar modelos capazes de suportar o armazenamento de toda a informação necessária, evitando e prevenindo determinados erros e falhas para assim conseguir desenvolver algo fiável, importante e frágil como uma base de dados.

1.4. Estrutura do Relatório

Este relatório é composto por capítulos e subcapítulos que exploram cada passo realizado neste trabalho. A estrutura do mesmo está de acordo com o modelo da metodologia abordada no livro Database Systems (A Pratical Approach to Design, Implementation, and Management - 6th edition).

Como principais capítulos temos o Desenho Conceptual da Base de Dados, o Desenho Lógico da Base de Dados para o Modelo Relacional, o Desenho do Modelo Físico e por fim as Conclusões e Trabalhos futuros.

Em cada um destes capítulos teremos subcapítulos referentes a cada etapa necessária para responder ao modelo da metodologia adotada referida acima.

2. Desenho conceptual da base de dados

Analisado o enunciado do trabalho prático iniciamos a 1ª fase, o desenho conceptual da base de dados. Nesta fase contemplamos os seguintes passo:

2.1. Identificação de entidades
2.2. Identificação de relacionamentos
2.3. Identificação e associação de atributos a tipos de entidade ou relacionamento
2.4. Determinação do domínio de atributos
2.5. Determinação dos atributos de chave candidata, primária e alternativa
2.6. Verificação de redundância no modelo
2.7. Validação do modelo de dados conceptual em relação às transações do utilizador

2.1. Identificação de entidades

Dado o problema proposto encontramos as seguintes entidades:

Nome da Entidade	Descrição	Apelido	Ocorrência
Contacto	Termo geral para uma lista de contactos		A app tem uma lista de contactos
Chat	Termo geral para descrição de uma conversa		Um contacto envia mensagens para um chat
Mensagem	Termo geral para uma mensagem enviada por um contacto		Um contacto envia uma mensagem
Histórico	Termo geral para o histórico de todas as mensagens enviadas por determinado contacto para determinado chat		Um contacto envia uma mensagem para um chat

Tabela 1 - Identificação de Entidades

2.2. Identificação de relacionamentos

Após identificadas e documentadas as entidades para a resolução do problema existe a necessidade de relacionar as mesmas. Para tal é apresentada a seguinte tabela com a identificação da entidade com a respetiva multiplicidade, o nome da relação e a entidade com que esta de relaciona na multiplicidade correspondente.

Entidade	Multiplicidade	Relação	Multiplicidade	Entidade
Contacto	1.. * 1	Participa Está	1.. * 0.. *	Chat Histórico
Chat	1 1.. *	Está Contém	0.. * 1.. *	Histórico Contacto
Mensagem	1	Existe	0.. *	Histórico
Histórico	0..* 0..* 0.. *	Tem Existe Pertencem	1 1 1	Contacto Mensagem Chat

Tabela 2 - Identificação de Relacionamentos

Perante todas as informações obtidas até este ponto podemos estruturar as entidades e seus relacionamentos num Desenho Relacional da base de dados.

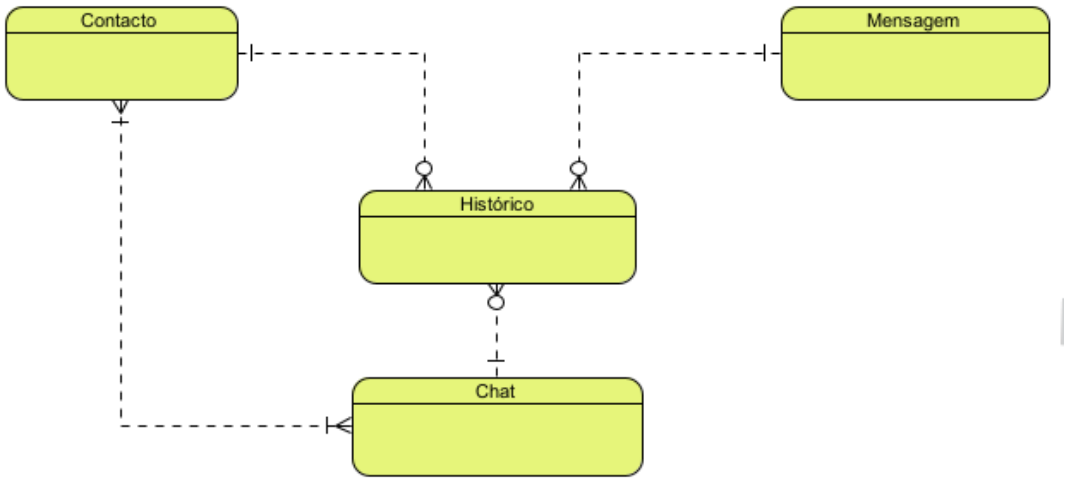

Figura 1 - Diagrama Conceptual

2.3. Identificação e associação de atributos a tipos de entidade ou relacionamento

Depois de encontradas as entidades e suas relações chegamos ao ponto da identificação dos atributos para a respetiva entidade.

Entidade	Atributos	Descrição
Contactos	N_Telemóvel	Número de telemóvel
	Nome	Nome da pessoa a quem pertence o contacto
	Email	Endereço de correio eletrónico
	Status	String que identifica se o contacto está ou não bloqueado
Chat	Chat_id	Código que identifica o chat
	nome	Nome do chat
	N_Telemóvel	Número de telemóvel de cada participante
	Tipo	Caracter que identifica o tipo de grupo (I-Individual, G-Grupo)
	Status	String que identifica se o chat está bloqueado
	Time	Tempo que o chat está bloqueado
	Numero_Partcipantes	Número de participantes do chat
Mensagem	Mensagem_Id	Código numérico que identifica a mensagem
	Text	Texto da mensagem
Histórico	History_ID	Código numérico que identifica o histórico
	Remetente_Contacto	Número de telemóvel do remetente
	Chat_Id	Código que identifica o chat para onde foi enviada
	Mensagem_Id	Código que identifica a mensagem que foi enviada
	Data	Data de quando foi enviada a mensagem
	Hora	Hora de quando foi enviada a mensagem

Tabela 3 - Identificação de atributos a entidades ou relacionamentos

2.4. Determinação do domínio de atributos

Dado termos encontrado atributos para corresponder a uma solução do problema é essencial a criação de restrições dos atributos para uma base de dados com tabelas concisas. Deste modo, apresentamos em seguida a caracterização do domínio de cada atributo.

Atributo	Tipo e tamanho	Nulos	Múltiplos Valores	Derivado	Composto
Contactos					
N_Telemóvel	Número inteiro de 9 dígitos	N	N	N	N
Nome	Conjunto de 30 caracteres	N	N	N	N
Email	Conjunto de 30 caracteres	N	N	N	N
Status	Conjunto de 12 carateres	N	N	N	N
Chat					
Chat_Id	Número inteiro	N	N	N	N
Nome	Conjunto de 30 caracteres	N	N	N	N
Status	Conjunto de 12 carateres	N	N	N	N
Time	Hora HH:MM:SS	N	N	N	N
Tipo	1 Caracter (I,G)	N	N	N	N
Numero_Participantes	Número inteiro de 4 dígitos	N	N	N	N
Mensagem					
Mensagem_Id	Número inteiro	N	N	N	N
Texto	conjunto de 3000 caracteres	N	N	N	N
Histórico					
Histórico_Id	número inteiro	N	N	N	N
Chat_Id	Número inteiro	N	N	N	N
Remetente_contacto	Número inteiro de 9 dígitos	N	N	N	N
Mensagem_Id	Conjunto de 20 caracteres	N	N	N	N
Data	Data YYYY-MM-DD	N	N	N	N
Hora	Hora HH:MM:SS	N	N	N	N

Tabela 4 - Identificação do domínio dos atributos

2.5. Determinação dos atributos de chave candidata, primária e alternativa

Apresentamos a proposta de chaves primárias:

Entidade	Chaves Candidatas
Contactos	N_Telemóvel, Email
Chat	Chat_Id,Nome
Mensagem	Mensagem_Id,Texto
Histórico	History_Id, (Remetente_contacto,Chat_Id,Mensagem_Id,Data,Hora)

Tabela 5 - Determinação das chaves candidatas

Entidade	Chave Primária
Contactos	N_telemóvel
Chat	Chat_Id
Mensagem	Mensagem_id
Histórico	History_Id

Tabela 6 - Determinação das chaves primárias

Entidade	Chave Secundária
Contactos	Email
Chat	Nome
Mensagem	Texto
Histórico	(Remetente_contacto, Chat_Id, Mensagem_Id, Data, Hora)

Tabela 7 - Determinação das chaves secundárias

Neste ponto estamos em condições de apresentar o seguinte desenho conceptual:

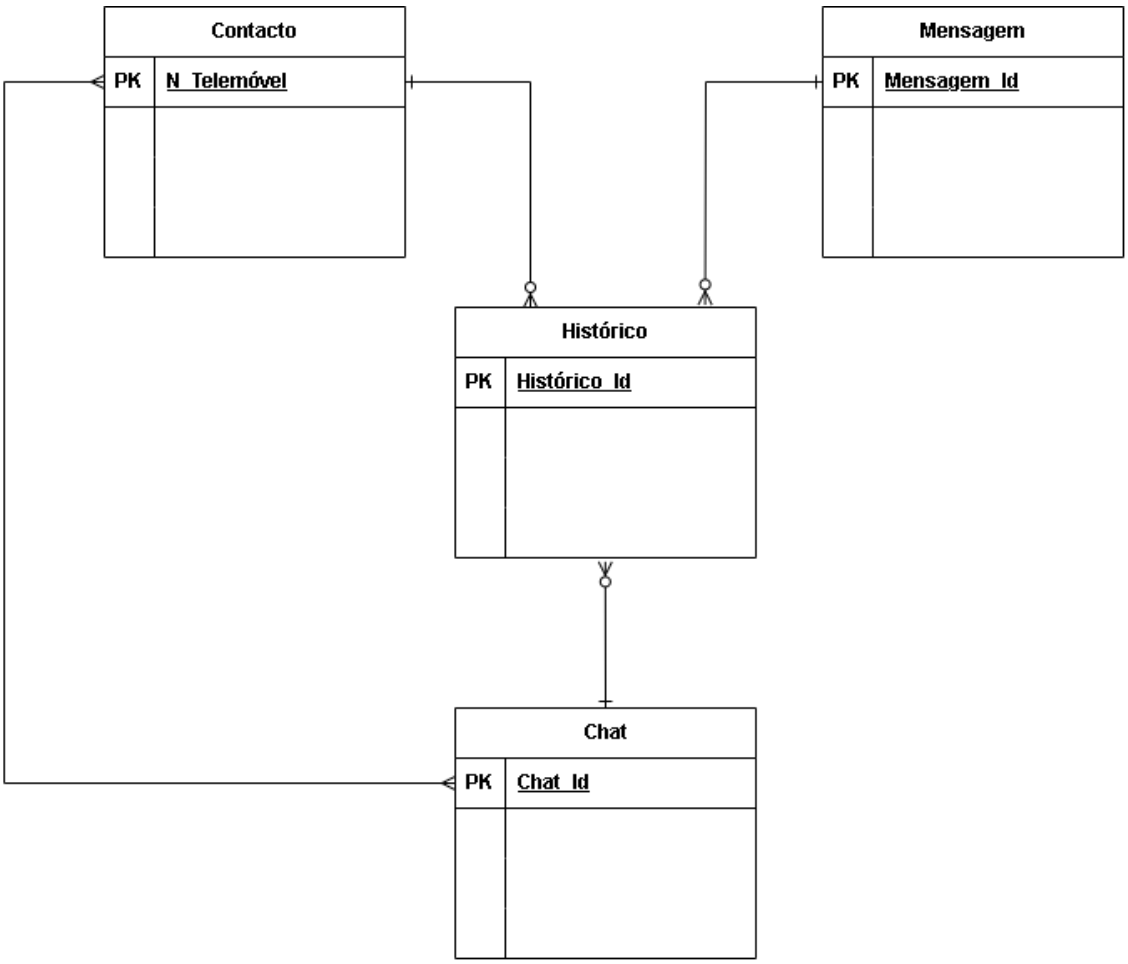

Figura 2. Diagrama Conceptual com chaves primárias

2.6. Verificação de redundância no modelo

Para a verificação de redundância no modelo até então construído seguimos as atividades:

a) Reexaminar das relações um para um

Data a inexistência de relacionamentos de um para um, duas entidades relacionam-se entre sim com multiplicidade 1 em ambas, não temos nada a revalidar nesta atividade.

b) Remover relações redundantes

Um relacionamento é redundante se a mesma informação poder ser obtida por outro relacionamento. [1] Assim sendo não encontramos redundância no modelo apresentado anteriormente.

c) Considerar a dimensão de tempo

A dimensão temporal dos relacionamentos é importante ao avaliar a redundância. [1] Como não foi detetada redundância nos relacionamentos então, não efetuamos a consideração sobre a dimensão temporal.

2.7. Validação do modelo de dados conceptual em relação às transações do utilizador

O objetivo desta etapa é verificar o modelo para garantir que o modelo suporta as transações necessárias. Usando o modelo, tentamos realizar as operações manualmente. Se pudermos resolver todas as transações dessa maneira, verificamos que o modelo de dados conceptual suporta as transações necessárias. No entanto, se não conseguirmos realizar uma transação manualmente, deve haver um problema com o modelo de dados, que deve ser resolvido. Nesse caso, é provável que tenhamos omitido uma entidade, um relacionamento ou um atributo do modelo de dados. [1]

Perante isto o modelo é válido e podemos seguir para a próxima etapa.

3. Desenho lógico da base de dados para o Modelo Relacional

Seguimos para a 2ª fase, o desenho lógico, onde realizamos os seguintes tópicos:

3.1. Derivação de relações para modelo de dados lógico

3.2. Validação de relações com recurso a normalização

3.3. Validação de relações em relação às transações do utilizador

3.4. Validação de restrições de integridade

3.5. Revisão do modelo de dados lógicos com o utilizador

3.6. Verificação do provável crescimento futuro

3.1. Derivação de relações para modelo de dados lógico

Nesta etapa, derivamos relações para o modelo de dados lógico para representar as entidades, relacionamentos e atributos. Descrevemos a composição de cada relação usando uma DBDL para BD relacionais. Utilizando a DBDL, primeiro especificamos o nome da relação, seguido por uma lista dos atributos simples da relação entre parenteses.

Em seguida, identificamos a chave primária e qualquer chave(s) alternativa(s) e/ou estrangeira(s) da relação. Após a identificação de uma chave estrangeira, é fornecida a relação que contém a chave primária referenciada. Quaisquer atributos derivados também são listados, juntamente com a forma como cada um é calculado. [1]

Contactos (N_Telemóvel, Nome, email, Status)

CP N_Telemóvel

CA Email

Chat (Chat_Id, Nome, Tipo, Status, Time)

CP Chat_Id

CA Nome

Mensagem (Mensagem_id, Texto)

CP Mensagem_Id

CA Texto

Histórico (Histórico_Id, N_Telemóvel, Chat_Id, Mensagem_Id, Data, Hora)

CP Histórico_id

CA Remetente_contacto, Chat_Id, Mensagem_Id, Data, Hora

CE N_Telemóvel

CE Chat_Id

CE Mensagem_Id

3.2. Validação de relações com recurso a normalização

A normalização é uma técnica formal para analisar relações com base na sua chave primária (ou chaves candidatas) e dependências funcionais (Codd, 1972b). [1] O processo de normalização pode ser resumido pelo diagrama:

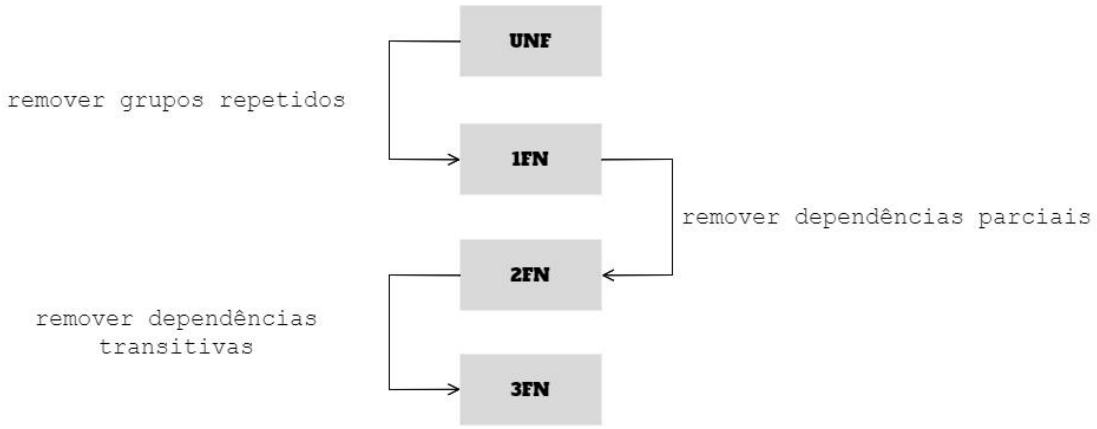

Figura 3 - Ilustração em diagrama do processo de normalização (adaptado Figure 14.8 [1])

Contacto

» UNF

Telefone	Nome	Email	Status
912365	Manuela	love@gmail.com	Bloqueado
18	Fábio	biofa@gmail.com	Desbloqueado

Tabela 8 - Exemplo tabela 'Cliente' na UNF

Não é necessário fazer nenhuma modificação, pois cada atributo já contém apenas valores atômicos e não repetidos.

Historico

» UNF

Historico_Id	Remetente_Contacto	Chat_Id	Mensagem_Id	Data	Hora
6	932250183	1	1	2023-07-08	21:22:50
8	932250198	1	2	2023-07-09	13:37:52.

Tabela 9 - Exemplo tabela 'Historico' na UNF

Não é necessário fazer nenhuma modificação, pois cada atributo já contém apenas valores atômicos e não repetidos.

Chat

» UNF

Chat_Id	Nome	Tipo	Status	Numero_Participantes
01	Chat Privado	P	Desbloqueado	2
02	Grupo de Amigos	G	Desbloqueado	4

Tabela 10 - Exemplo tabela 'Chat' na UNF

Não é necessário fazer nenhuma modificação, pois cada atributo já contém apenas valores atômicos e não repetidos.

Participantes

» UNF

Contacto	Chat_Id
912349876	2
912349876	7
932250183	1
932250198	1

Tabela 11 - Exemplo tabela 'Participantes' na UNF

Não é necessário fazer nenhuma modificação, pois cada atributo já contém apenas valores atômicos e não repetidos.

Mensagem

» UNF

Mensagem_Id	Texto
1	Olá, como você está?
2	Estou bem, obrigado! E você?
3	Temos uma reunião marcada para amanhã.

Tabela 12 - Exemplo tabela ''Mensagem'' na UNF

Não é necessário fazer nenhuma modificação, pois cada atributo já contém apenas valores atômicos e não repetidos.

3.3. Validação de relações em relação às transações do utilizador

Neste passo verificamos com o cliente as transações que este espera poder realizar aquando da prontificação da BD. Recordamos que em Apresentação do Caso de Estudo foram definidas as transações do utilizador:

- Qual o contacto para o qual são enviadas mais mensagens.
- Liste o número de mensagens recebidas por semana relativamente ao ano de 2022.
- Lista dos grupos de contactos, ordenada por ordem descendente da quantidade de participantes no grupo.
- Qual a hora do dia em que envia/recebe mais mensagens?
- Outras que achar relevantes.

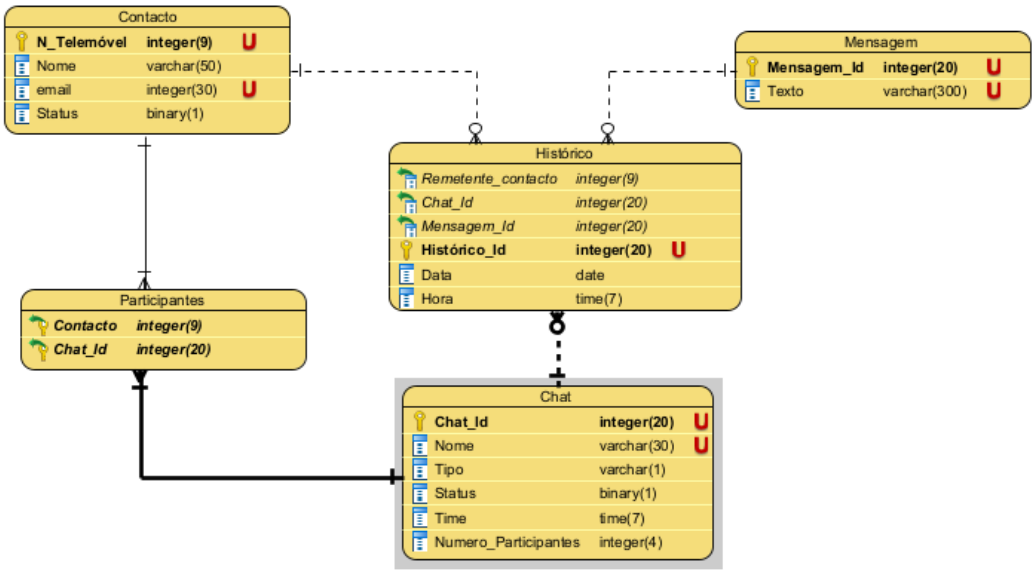

Figura 4 - Diagrama Lógico

Com o Diagrama Lógico realizado até então somos capazes de conseguir responder às questões do utilizador.

3.4. Validação de restrições de integridade

Restrições de integridade são as restrições que desejamos impor para proteger a base de dados de se tornar incompleta, imprecisa ou inconsistente. [1]

No caso em estudo dá se importância às seguintes restrições:

- O primeiro contacto da tabela Contacto é a pessoa que detém o dispositivo. Ao instalar a aplicação e inserir os dados fica automaticamente na primeira posição da tabela.
- Na tabela Mensagem, caso escreva um texto que já exista na tabela este é auto completado, de forma a não criar mais um tuplo na tabela. Isto permite também a aplicação futuramente expandir para mensagens automáticas ou pré-definidas. Esta abordagem reduz drasticamente a redundância de dados na tabela Mensagem. Caso esteja a criar um texto que ainda não existe será adicionado à tabela.

3.5. Revisão do modelo de dados lógicos com o utilizador

Após a execução de todos os passos constatamos que o modelo de dados lógico está completo e totalmente documentado.

Apresentamos diagrama lógico finalizado:

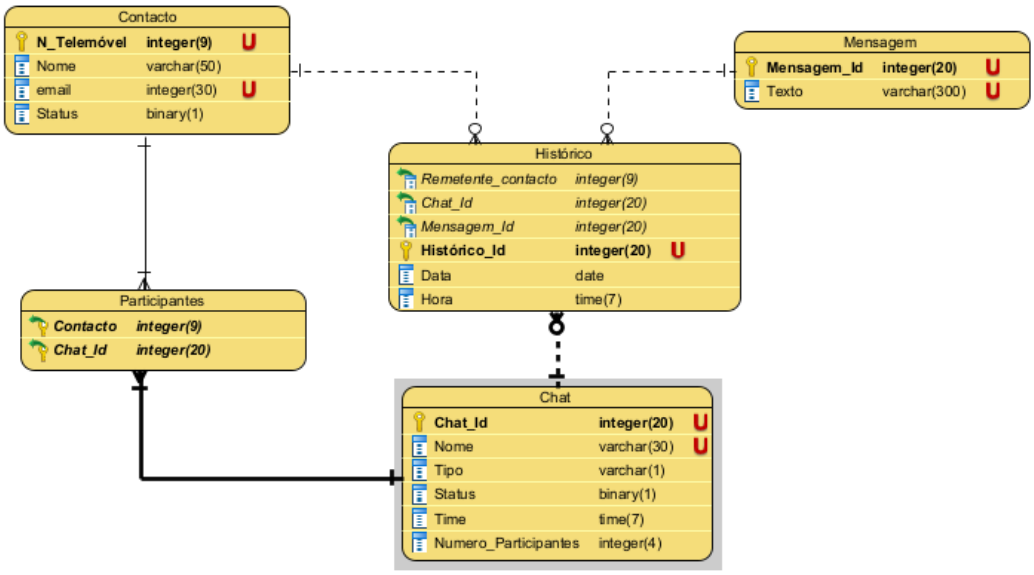

Figura 5 - Diagrama Lógico validado

3.6. Verificação do provável crescimento futuro

A BD apresentada em Revisão do modelo de dados lógicos com o utilizador pode futuramente ser expandida e incrementadas novas tabelas. A expansão futura pode ser realizada reaproveitando a informação já existente.

Exemplos de expansão futura poderão ser o incremento de chats de outro tipo, por exemplo temporários (expiram ao final de determinado tempo), também adicionar a possibilidade de enviar outras mensagens sem ser texto, imagens ou áudios e também mensagens pré-programadas.

4. Desenho do Modelo Físico

Concluímos o processo com a 3ª fase, o desenho do modelo físico, seguindo os passos:

4.1. Tradução no Modelo Lógico para o SGBD

4.2. Desenho das vistas de utilizador

4.1. Tradução do Modelo Lógico para o SGBD

Neste passo introduzimos o modelo lógico no SGBD escolhido Microsoft SQL Server

Começamos pela criação das tabelas e consequente relação entre elas.

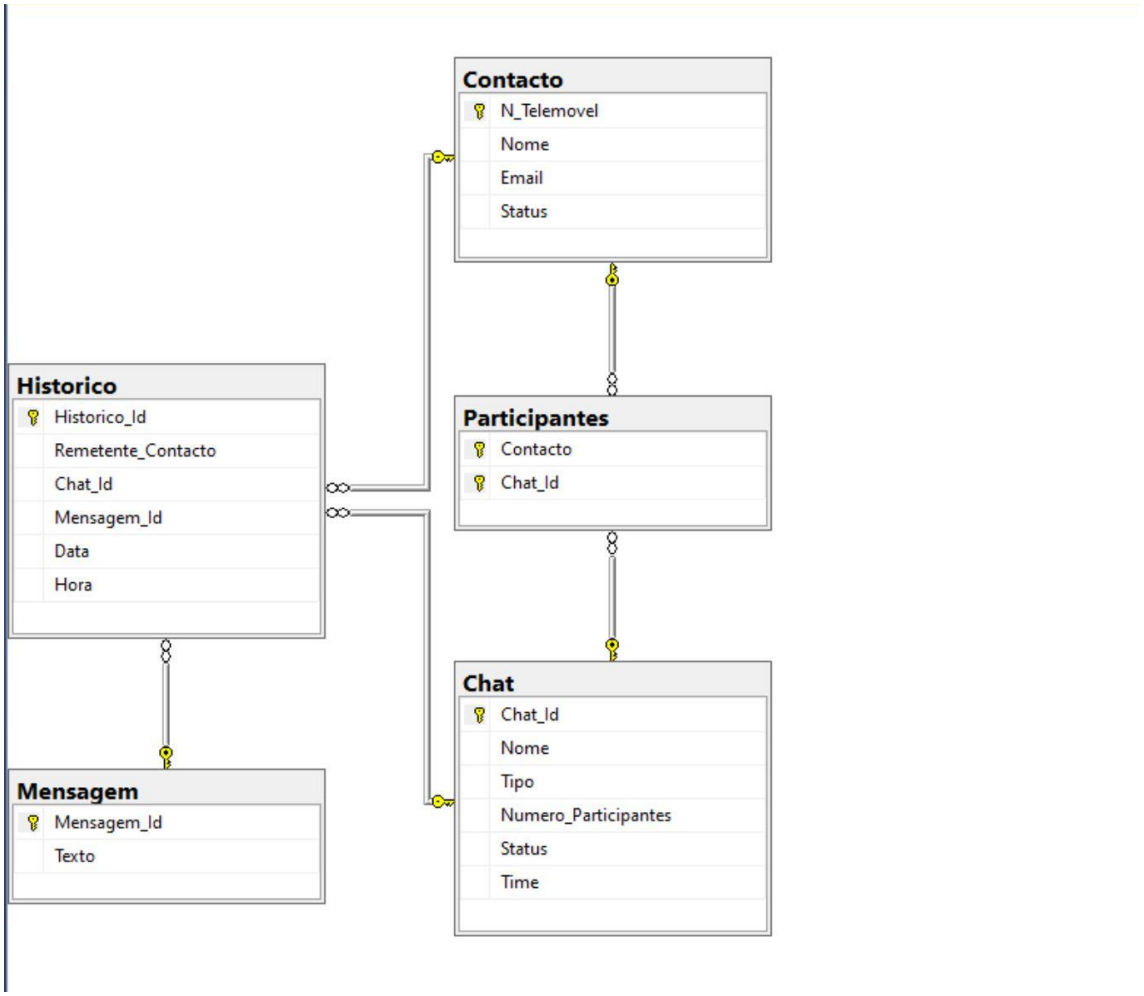

Figura 6 - Diagrama Físico

4.2. Desenho das vistas de utilizador

Para dar resposta ao pretendido conseguimos as seguintes vistas:

- Qual o contacto para o qual são enviadas mais mensagens.

```
SELECT TOP 1 c.N_Telemovel, COUNT(*) AS TotalMensagensRecebidas
FROM Historico h
JOIN Participantes p ON h.Chat_Id = p.Chat_Id
JOIN Contacto c ON p.Contacto = c.N_Telemovel
GROUP BY c.N_Telemovel
ORDER BY TotalMensagensRecebidas DESC;
```

- Liste o número de mensagens recebidas por semana relativamente ao ano de 2022.

```
SELECT DATEPART(WEEK, Data) AS Semana, COUNT(*) AS NumeroMensagens
FROM Historico
WHERE YEAR(Data) = 2022
GROUP BY DATEPART(WEEK, Data);
```

- Lista dos grupos de contactos, ordenada por ordem descendente da quantidade de participantes no grupo.

```
SELECT c.Chat_Id, c.Nome, COUNT(p.Contacto) AS NumeroParticipantes
FROM Chat c
LEFT JOIN Participantes p ON c.Chat_Id = p.Chat_Id
WHERE c.Tipo = 'G'
GROUP BY c.Chat_Id, c.Nome
ORDER BY COUNT(p.Contacto) DESC;
```

- Qual a hora do dia em que envia/recebe mais mensagens?

```
SELECT CASE
    WHEN DATEPART(HOUR, Hora) = 0 THEN '00:00'
    ELSE CONVERT(VARCHAR, DATEPART(HOUR, Hora)) + ':00'
   END AS Hora, COUNT(*) AS NumeroMensagens
FROM Historico
GROUP BY DATEPART(HOUR, Hora)
ORDER BY COUNT(*) DESC;
```

5. Decisões tomadas e Conclusões

Decidimos criar uma base de dados para um sistema de mensagens que permite a troca de mensagens entre contactos em chats privados e em grupos. Durante o processo de criação, tomámos algumas decisões importantes para garantir a eficiência, a integridade dos dados e satisfazer os requisitos do sistema.

Começámos por definir as tabelas principais. A tabela "Contacto" armazena as informações dos contactos, como número de telemóvel, nome, email e estado de bloqueio. Optámos por utilizar o número de telemóvel como chave primária, pois é uma informação única e fundamental para identificar cada contacto. Também aplicámos validações para garantir que o número de telemóvel está dentro de um intervalo válido e que o email tem um formato adequado.

Em seguida, criámos a tabela "Chat", que representa os chats onde as mensagens são trocadas. Definimos o Chat_Id como chave primária, permitindo a identificação única de cada chat. Adicionámos campos para o nome do chat, tipo (privado ou em grupo), número de participantes e estado de bloqueio. Decidimos utilizar um campo para registar a última atualização do estado do chat, o que pode ser útil para acompanhar as alterações ao longo do tempo.

Para associar os contactos aos chats, criámos a tabela "Participantes". Esta tabela possui duas chaves estrangeiras, referenciando o número de telemóvel do contacto e o Chat_Id do chat correspondente. Dessa forma, garantimos que um contacto não possa participar do mesmo chat mais de uma vez.

Para armazenar as mensagens trocadas nos chats, criámos a tabela "Mensagem". Cada mensagem possui um identificador único (Mensagem_Id) e o texto da mensagem. Esta tabela será fundamental para registar o conteúdo das conversas.

Por fim, implementámos a tabela "Histórico", que regista o histórico de mensagens enviadas nos chats. Esta tabela possui chaves estrangeiras para o remetente da mensagem, o Chat_Id correspondente e o Mensagem_Id da mensagem enviada. Além disso, incluímos campos para registar a data e a hora do envio da mensagem. Estas informações serão úteis para rastrear as interações no sistema e realizar análises temporais.

Ao projetar as tabelas, também considerámos a integridade dos dados. Implementámos diversas restrições de integridade, como chaves primárias, chaves estrangeiras e verificações de intervalo, formato e valores permitidos. Estas restrições garantem que os dados armazenados estejam consistentes e em conformidade com as regras estabelecidas.

Além das tabelas, implementámos uma série de triggers para adicionar lógica adicional aos eventos de inserção, atualização e exclusão nas tabelas. Os triggers foram projetados para realizar validações, como impedir a adição de participantes duplicados no mesmo chat, impedir a inclusão de remetentes bloqueados no histórico e atualizar automaticamente o número de participantes num chat.

No geral, a nossa abordagem para a criação desta base de dados foi cuidadosa e focada em atender aos requisitos do sistema. Procurámos garantir a integridade dos dados, a eficiência das operações e a consistência das informações registadas. Acreditamos que esta estrutura seja sólida e capaz de suportar um sistema de mensagens confiável e eficaz.

Na sequência, foi desenvolvida as 'views', a fim de facilitar as consultas mais corriqueiras implementadas anteriormente, desta forma, os gestores do sistema poupam tempo necessário para tal.

Concluímos assim o trabalho prático com sucesso na medida em que o realizado corresponde às necessidades solicitadas.
Consolidamos os conhecimentos da Unidade Curricular com a execução em trabalho colaborativo e de pequena escala, mas ajustado à realidade fora do contexto académico.
Futuramente, podia ser realizada a integração de um módulo financeiro na BD de forma a torná-la mais rica e maleável.

Bibliografia

[1] C. E. B. THOMAS M. CONNOLLY, DATABASE SYSTEMS - A Practical Approach to Design, Implementation, and Management, Pearson, 2015.

Referências WWW

[01] **https://moodle2.estg.ipp.pt/course/view.php?id=214**

Página do Moodle da Unidade Curricular onde encontramos os conteúdos sobre BD.

[02] **https://www.w3schools.com/sql/default.asp**

Página da organização W3Schools onde encontramos alguma documentação e exemplos ações a executar em BD.

[03] **https://www.dcc.fc.up.pt/~edrdo/aulas/bd/teoricas/**

Página do Departamento de Ciências de Computadores da Faculdade de Ciências da Universidade do Porto na Unidade Curricular de Base de Dados 2021/22 que tem disponível conteúdos de BD.

Lista de Siglas e Acrónimos

SGBD	Sistema de Gestão de Base de Dados
BD	Base de Dados
DBDL	Database Disignation Language (Linguagem de designação de base de dados)
CP	Chave Primária
CA	Chave Alternativa
CE	Chave Estrangeira
UNF	Forma não normalizada
1FN	1ª forma normalizada
2FN	2ª forma normalizada
3FN	3ª forma normalizada

Anexos

Create | Código para criar todas as tabelas

```
CREATE TABLE Contacto (
   N_Telemovel INT PRIMARY KEY CHECK (N_Telemovel >= 100000000 AND N_Telemovel <= 999999999),
   Nome VARCHAR(30) NOT NULL,
   Email VARCHAR(30) NOT NULL CHECK (email LIKE '%@%.%'),
   Status VARCHAR(12) NOT NULL DEFAULT 'Desbloqueado' CHECK (status IN ('Desbloqueado', 'Bloqueado')),
);


CREATE TABLE Chat (
   Chat_Id INT PRIMARY KEY IDENTITY(1,1),
   Nome VARCHAR(30) NOT NULL,
   Tipo VARCHAR(1) CHECK (Tipo IN ('P', 'G')), -- Restrição para permitir apenas os valores 'P' (para chat privado) e 'G' (para chat em grupo)
Numero_Participantes INT DEFAULT 0,
   Status VARCHAR(12) NOT NULL DEFAULT 'Desbloqueado' CHECK (status IN ('Desbloqueado', 'Bloqueado')),
Time Time(7)
);


CREATE TABLE Participantes (
   Contacto INT,
   Chat_Id INT,
   PRIMARY KEY (Contacto, Chat_Id),
   FOREIGN KEY (Contacto) REFERENCES Contacto(N_telemovel) ON UPDATE CASCADE ON DELETE CASCADE,
   FOREIGN KEY (Chat_Id) REFERENCES Chat(Chat_Id) ON UPDATE CASCADE ON DELETE CASCADE,
);


-- Criação da tabela Mensagem
CREATE TABLE Mensagem (
```

```
    Mensagem_Id INT PRIMARY KEY IDENTITY (1,1),
    Texto VARCHAR(3000) NOT NULL
);


CREATE TABLE Historico (
    Historico_Id INT PRIMARY KEY IDENTITY (1,1),
    Remetente_Contacto INT,
    Chat_Id INT,
    Mensagem_Id INT,
    Data DATE,
    Hora TIME,
    FOREIGN KEY (Remetente_Contacto) REFERENCES Contacto(N_telemovel) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (Chat_Id) REFERENCES Chat(Chat_Id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (Mensagem_Id) REFERENCES Mensagem(Mensagem_Id) ON UPDATE CASCADE ON DELETE CASCADE,
    CHECK (Data IS NOT NULL),
    CHECK (Hora IS NOT NULL)
);
```

Trigger | Todos os triggers do trabalho

```
CREATE TRIGGER atualizar_numero_participantes
ON Participantes
AFTER INSERT, DELETE
AS
BEGIN
    -- Atualizar o número de participantes para cada Chat_Id afetado pelo trigger
    UPDATE Chat
    SET Numero_Participantes = (SELECT COUNT(*) FROM Participantes WHERE Chat_Id = Chat.Chat_Id)
    WHERE Chat_Id IN (SELECT Chat_Id FROM inserted UNION SELECT Chat_Id FROM deleted);
END;

CREATE TRIGGER atualizar_time_chat ON Chat
AFTER UPDATE
```

```
AS
BEGIN
   IF UPDATE(Status)
   BEGIN
      UPDATE c
      SET c.Time = CASE
         WHEN i.Status = 'Bloqueado' THEN CAST('23:59:59' AS TIME)
         WHEN i.Status = 'Desbloqueado' THEN CAST('00:00:00' AS TIME)
         END
      FROM Chat c
      INNER JOIN inserted i ON c.Chat_Id = i.Chat_Id;
   END
END;

CREATE TRIGGER limite_participantes_privados
ON Participantes
AFTER INSERT, UPDATE
AS
BEGIN
   DECLARE @chat_id INT;
   DECLARE @num_participantes INT;

   SELECT @chat_id = Chat_Id
   FROM inserted;

   SELECT @num_participantes = COUNT(*)
   FROM Participantes
   WHERE Chat_Id = @chat_id;

   IF EXISTS (SELECT 1 FROM Chat WHERE Chat_Id = @chat_id AND Tipo = 'P' AND @num_participantes >
2)
   BEGIN
      RAISERROR ('Um chat privado não pode ter mais de 2 participantes.', 16, 1);
      ROLLBACK TRANSACTION;
   END;
END;
```

```
CREATE TRIGGER impedir_participante_bloqueado
ON Participantes
FOR INSERT
AS
BEGIN
   -- Verificar o status do contato
   DECLARE @contato_status VARCHAR(12);
   SELECT @contato_status = c.Status
   FROM Contacto c
   INNER JOIN inserted i ON c.N_Telemovel = i.Contacto;

   -- Se o status for bloqueado, cancelar a inserção do participante
   IF @contato_status = 'Bloqueado'
   BEGIN
      RAISERROR ('Não é possível adicionar um participante com o contato bloqueado.', 16, 1);
      ROLLBACK TRANSACTION;
   END;
END;

CREATE TRIGGER validar_contato_existente
ON Participantes
FOR INSERT
AS
BEGIN
   -- Verificar a existência dos contatos
   DECLARE @num_participantes INT;
   SELECT @num_participantes = COUNT(*)
   FROM inserted i
   LEFT JOIN Contacto c ON c.N_Telemovel = i.Contacto
   WHERE c.N_Telemovel IS NULL;

   -- Se houver contatos inexistentes, cancelar a inserção dos participantes
   IF @num_participantes > 0
   BEGIN
      RAISERROR ('Não é possível adicionar participantes com contatos inexistentes.', 16, 1);
      ROLLBACK TRANSACTION;
   END;
END;
```

```
CREATE TRIGGER bloquear_participantes_chat
ON Participantes
FOR INSERT
AS
BEGIN
   -- Verificar o status do chat
   DECLARE @status_chat VARCHAR(12);
   SELECT @status_chat = c.Status
   FROM inserted i
   JOIN Chat c ON c.Chat_Id = i.Chat_Id;

   -- Se o status do chat for bloqueado, cancelar a inserção dos participantes
   IF @status_chat = 'Bloqueado'
   BEGIN
      RAISERROR ('Não é possível adicionar participantes em um chat bloqueado.', 16, 1);
      ROLLBACK TRANSACTION;
   END;
END;

CREATE TRIGGER verificar_numero_telefone
ON Contacto
INSTEAD OF INSERT
AS
BEGIN
   IF EXISTS (SELECT 1 FROM Contacto WHERE N_Telemovel IN (SELECT N_Telemovel FROM inserted))
   BEGIN
      RAISERROR ('Já existe um contato com o mesmo número de telefone.', 16, 1);
      ROLLBACK TRANSACTION;
   END;
   ELSE
   BEGIN
      INSERT INTO Contacto (N_Telemovel, Nome, Email, Status)
      SELECT N_Telemovel, Nome, Email, Status
      FROM inserted;
   END;
END;
```

```
CREATE TRIGGER verificar_participante_duplicado
ON Participantes
AFTER INSERT
AS
BEGIN
  IF EXISTS (
    SELECT 1
    FROM inserted i
    JOIN Participantes p ON p.Contacto = i.Contacto AND p.Chat_Id = i.Chat_Id
  )
  BEGIN
    RAISERROR ('O participante já está inserido no chat.', 16, 1);
    ROLLBACK TRANSACTION;
  END;
END;


CREATE TRIGGER verificar_remetente_bloqueado ON Historico
INSTEAD OF INSERT
AS
BEGIN
  SET NOCOUNT ON;

  DECLARE @contato_status VARCHAR(12);
  SELECT @contato_status = c.Status
  FROM inserted i
  INNER JOIN Contacto c ON c.N_Telemovel = i.Remetente_Contacto;

  IF @contato_status = 'Bloqueado'
  BEGIN
    RAISERROR ('Não é possível adicionar um remetente bloqueado à tabela Historico.', 16, 1);
  END;
  ELSE
  BEGIN
    INSERT INTO Historico (Remetente_Contacto, Chat_Id, Mensagem_Id, Data, Hora)
    SELECT i.Remetente_Contacto, i.Chat_Id, i.Mensagem_Id, i.Data, i.Hora
```

```
    FROM inserted i;
  END;
END;

CREATE TRIGGER verificar_chat_bloqueado ON Historico
AFTER INSERT
AS
BEGIN
  SET NOCOUNT ON;

  DECLARE @chat_status VARCHAR(12);
  SELECT @chat_status = c.Status
  FROM inserted i
  INNER JOIN Chat c ON c.Chat_Id = i.Chat_Id;

  IF @chat_status = 'Bloqueado'
  BEGIN
    RAISERROR ('Não é possível adicionar registros à tabela Historico para um chat bloqueado.', 16, 1);
    DELETE FROM Historico WHERE Historico_Id IN (SELECT Historico_Id FROM inserted);
  END;
END;



CREATE TRIGGER verificar_participante_chat ON Historico
AFTER INSERT
AS
BEGIN
  SET NOCOUNT ON;

  IF EXISTS (
    SELECT 1
    FROM inserted i
    LEFT JOIN Participantes p ON p.Contacto = i.Remetente_Contacto AND p.Chat_Id = i.Chat_Id
    WHERE p.Chat_Id IS NULL
  )
  BEGIN
    RAISERROR ('Não é permitido adicionar um participante que não pertence ao Chat.', 16, 1);
```

```
        ROLLBACK TRANSACTION;
    END;
END;


CREATE TRIGGER excluir_chat_privado
ON Contacto
AFTER DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DELETE FROM Participantes
    WHERE Chat_Id IN (
        SELECT p.Chat_Id
        FROM Participantes p
        INNER JOIN deleted d ON p.Contacto = d.N_Telemovel
        INNER JOIN Chat c ON p.Chat_Id = c.Chat_Id
        WHERE c.Tipo = 'P'
    );

    DELETE FROM Chat
    WHERE Chat_Id IN (
        SELECT p.Chat_Id
        FROM Participantes p
        INNER JOIN deleted d ON p.Contacto = d.N_Telemovel
        INNER JOIN Chat c ON p.Chat_Id = c.Chat_Id
        WHERE c.Tipo = 'P'
    );
END;

CREATE TRIGGER verificar_participante_duplicado
ON Participantes
AFTER INSERT
AS
BEGIN
    IF EXISTS (
        SELECT Contacto, Chat_Id
```

```
    FROM inserted
    GROUP BY Contacto, Chat_Id
    HAVING COUNT(*) > 1
  )
  BEGIN
    RAISERROR('Não é permitido inserir o mesmo participante duas vezes no mesmo chat.', 16, 1);
    ROLLBACK TRANSACTION;
    RETURN;
  END;
END;

CREATE TRIGGER verificar_dados_duplicados
ON Historico
AFTER INSERT
AS
BEGIN
  IF EXISTS (
    SELECT Remetente_Contacto, Chat_Id, Mensagem_Id
    FROM Historico
    WHERE Historico_Id IN (
      SELECT Historico_Id
      FROM inserted
    )
    GROUP BY Remetente_Contacto, Chat_Id, Mensagem_Id
    HAVING COUNT(*) > 1
  )
  BEGIN
    RAISERROR('Não é permitido ter dados duplicados na tabela Historico.', 16, 1);
    ROLLBACK TRANSACTION;
    RETURN;
  END;
END;
```

Views | Todas as views do trabalho

```
CREATE VIEW ContatoMaisMensagens AS
SELECT TOP 1 Remetente_Contacto AS N_Telemovel, COUNT(*) AS TotalMensagens
FROM Historico
GROUP BY Remetente_Contacto
ORDER BY COUNT(*) DESC;

CREATE VIEW MensagensPorSemana AS
SELECT DATEPART(WEEK, Data) AS Semana, COUNT(*) AS NumeroMensagens
FROM Historico
WHERE YEAR(Data) = 2022
GROUP BY DATEPART(WEEK, Data);

CREATE VIEW GruposDeContatos AS
SELECT Chat_Id, Nome, NumeroParticipantes
FROM (
   SELECT c.Chat_Id, c.Nome, COUNT(p.Contacto) AS NumeroParticipantes,
        ROW_NUMBER() OVER (ORDER BY COUNT(p.Contacto) DESC) AS RowNumber
   FROM Chat c
   LEFT JOIN Participantes p ON c.Chat_Id = p.Chat_Id
   WHERE c.Tipo = 'G'
   GROUP BY c.Chat_Id, c.Nome
) AS subquery
WHERE RowNumber <= 100;



CREATE VIEW HoraMaisMensagens AS
SELECT TOP 100 PERCENT
   CASE
      WHEN DATEPART(HOUR, Hora) = 0 THEN '00:00'
      ELSE CONVERT(VARCHAR, DATEPART(HOUR, Hora)) + ':00'
   END AS Hora, COUNT(*) AS NumeroMensagens
```

```
FROM Historico
GROUP BY DATEPART(HOUR, Hora)
ORDER BY COUNT(*) DESC;



CREATE VIEW ContatosBloqueados AS
SELECT *
FROM Contacto
WHERE Status = 'Bloqueado';

CREATE VIEW MensagensPorContato AS
SELECT ct.N_Telemovel, ct.Nome, COUNT(*) AS TotalMensagens
FROM Historico h
INNER JOIN Contacto ct ON h.Remetente_Contacto = ct.N_Telemovel
GROUP BY ct.N_Telemovel, ct.Nome;

CREATE VIEW ContatosSemChat AS
SELECT ct.N_Telemovel, ct.Nome
FROM Contacto ct
LEFT JOIN Participantes p ON ct.N_Telemovel = p.Contacto
WHERE p.Contacto IS NULL;

CREATE VIEW MensagensRecebidasPorSemana AS
SELECT c.N_Telemovel, c.Nome, DATEPART(WK, h.Data) AS Semana, COUNT(h.Mensagem_Id) AS
NumeroMensagensRecebidas
FROM Contacto c
INNER JOIN Participantes p ON c.N_Telemovel = p.Contacto
INNER JOIN Historico h ON p.Chat_Id = h.Chat_Id
WHERE YEAR(h.Data) = 2022
GROUP BY c.N_Telemovel, c.Nome, DATEPART(WK, h.Data);

CREATE VIEW MensagensEnviadasPorContatoBloqueado AS
SELECT c.N_Telemovel, c.Nome, COUNT(h.Mensagem_Id) AS TotalMensagensEnviadas
FROM Contacto c
LEFT JOIN Historico h ON c.N_Telemovel = h.Remetente_Contacto
WHERE c.Status = 'Bloqueado'
GROUP BY c.N_Telemovel, c.Nome;
```

```
CREATE VIEW ViewTotalMensagensRecebidas AS
SELECT TOP 1 c.N_Telemovel, COUNT(*) AS TotalMensagensRecebidas
FROM Historico h
JOIN Participantes p ON h.Chat_Id = p.Chat_Id
JOIN Contacto c ON p.Contacto = c.N_Telemovel
GROUP BY c.N_Telemovel
ORDER BY TotalMensagensRecebidas DESC;

SELECT COUNT(*) AS NumeroMensagensRecebidas
FROM Historico h
JOIN Participantes p ON h.Chat_Id = p.Chat_Id
WHERE p.Contacto = <número do contato>;



SELECT * FROM ViewTotalMensagensRecebidas;
SELECT *FROM MensagensPorSemana;
SELECT *FROM GruposDeContatos;
SELECT *FROM HoraMaisMensagens;
SELECT *FROM ContatosBloqueados;
SELECT *FROM MensagensPorContato;
SELECT *FROM ContatosSemChat;
SELECT *FROM MensagensRecebidasPorSemana;
SELECT *FROM MensagensEnviadasPorContatoBloqueado;
```