Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

Disciplina: Base de Dados

Base de Dados para Transplante de Órgãos

Descrição do problema e modelagem

Daniel Umeda Kuhn 13676541
Gustavo Curado Ribeiro 14576732
Luís Filipe Silva Forti 14592348
Manoel Thomaz Gama da Silva Neto 13676392
Pedro Fuziwara Filho 13676840
Docente responsável: Profa. Elaine Parros Machado de Sousa

São Carlos
2º semestre / 2025

SUMÁRIO

1. Descrição do Problema e Requisitos de Dados

O sistema proposto está inserido no contexto das Comissões Intra-Hospitalares de Doação de Órgãos e Tecidos para Transplante (CIHDOTTs), que desempenham papel central na coordenação de transplantes de órgãos no Brasil. O objetivo é desenvolver uma base de dados que apoie todas as etapas do processo de transplante, desde a coleta até a recepção e utilização dos órgãos, garantindo maior eficiência, rastreabilidade e segurança.

A base de dados deverá contemplar informações completas sobre as pessoas envolvidas, incluindo médicos, enfermeiros, pacientes doadores e receptores. Para cada categoria, atributos específicos são armazenados: no caso dos profissionais de saúde, dados como registros de classe e especializações; nos pacientes, características clínicas relevantes como sexo biológico, peso, histórico médico, telefones de emergência e prioridade de transplante no caso dos receptores.

Além das pessoas, o sistema deve gerenciar os hospitais, que são responsáveis tanto pelos pacientes internados quanto pela equipe de funcionários em serviço em cada horário. Essa gestão permite acompanhar a capacidade de atendimento, considerando disponibilidade de leitos e salas cirúrgicas. Hospitais também podem conter laboratórios associados, responsáveis pela realização de exames essenciais (como histocompatibilidade e tipagem sanguínea). Apenas hospitais autorizados pelo Sistema Nacional de Transplantes (SNT) podem realizar os procedimentos, sendo necessário controlar o tipo de transplante permitido e a validade periódica dessa autorização.

A coleta e substituição dos órgãos requerem a participação de múltiplos médicos e enfermeiros da CIHDOTT. O doador e o receptor, estando no mesmo hospital, ainda requerem salas e, portanto, equipes médicas distintas.

Dessa forma, o sistema deve integrar informações sobre pessoas, hospitais, laboratórios, salas e procedimentos, permitindo o gerenciamento e a consulta de dados em todas as etapas. A base de dados se torna, assim, uma ferramenta estratégica para aumentar a eficiência operacional, redutoria de riscos e asseguradora da conformidade regulatória no processo de transplante de órgãos.

1.1. Objetivos do Sistema

O sistema busca:

- Garantir a rastreabilidade completa de cada transplante, desde a coleta do órgão até a cirurgia no receptor.
- Centralizar e organizar informações de pacientes, doadores, receptores, médicos, enfermeiros e hospitais.
- Controlar a disponibilidade e utilização de salas e equipamentos hospitalares.
- Permitir consultas eficientes que auxiliem na gestão da capacidade hospitalar, na seleção de doadores compatíveis e no histórico de procedimentos.
- Atender aos requisitos de segurança e privacidade, em conformidade com a Lei Geral de Proteção de Dados (LGPD), visto que se trata de dados clínicos sensíveis.

O público-alvo da aplicação são profissionais de saúde e gestores hospitalares envolvidos no processo de transplantes, especialmente os membros da CIHDOTT.

1.2. Entidades e Atributos Principais

No modelo de dados, a entidade Pessoa ocupa posição central, sendo identificada unicamente pelo CPF. Para cada pessoa registram-se informações gerais, como nome completo, endereço (que é composto por Estado, Cidade, Bairro, Rua e Número) e múltiplos números de telefone. A partir dessa entidade de base, distinguem-se especializações que definem os papéis desempenhados no processo: as pessoas podem atuar como Funcionários ou como Pacientes.

Entre os funcionários, incluem-se médicos e enfermeiros. Todo funcionário está vinculado a pelo menos um hospital, podendo atuar em mais de um. Os médicos possuem atributos específicos como o número de CRM único, enquanto os enfermeiros são identificados pelo COREN único, sendo que são também armazenadas como atributos as validades tanto do CRM quanto do COREN. Todos os médicos possuem uma única especialização, que deve ser escolhida dentre todas as especializações médicas reconhecidas no Brasil. Ambos médicos e enfermeiros exercem funções em cirurgias, mas apenas os médicos podem supervisionar exames laboratoriais, com um médico podendo supervisionar vários exames, mas um exame precisando ser supervisionado por apenas um médico.

Já a entidade Paciente contempla todas as pessoas que recebem tratamento hospitalar. Para cada paciente são armazenados dados clínicos relevantes, como sexo, data de nascimento, cor, peso, histórico médico, contatos de emergência e o registro de óbito, quando aplicável. O histórico médico é um atributo multivalorado que, para cada condição que o paciente teve, armazena-se o CID do diagnóstico, a data e a descrição da condição. Os pacientes podem ser classificados em duas subcategorias: Doadores e Receptores (pacientes que não têm interesse ou necessidade de trasplantes não são classificados em nenhum dos dois). Os receptores possuem prioridade associada à fila de transplantes para cada órgão, enquanto ambos doadores e receptores necessariamente relacionam-se aos tipos de órgãos que esperam receber ou doar (logo, quando a cirurgia de coleta/recepção é realizada, eliminam-se essas relações). O modelo também admite que uma mesma pessoa acumule mais de uma dessas categorias — por exemplo, ser ao mesmo tempo paciente doador e receptor — mas preserva a integridade lógica ao impedir que alguém seja doador e receptor do mesmo Tipo de Órgão.

A entidade Hospital também é fundamental. Cada hospital é distinguido por um código CNES único, e possui ainda CNPJ, nome fantasia, endereço e um conjunto de autorizações emitidas pelo Sistema Nacional de Transplantes (SNT). Essas autorizações, de validade temporal definida, especificam quais tipos de transplante a instituição está habilitada a realizar. Além disso, os hospitais mantêm vínculos diretos com seus funcionários e pacientes, de forma a garantir tanto o controle da equipe em serviço quanto a gestão da capacidade de internação. São salvos também, como atributos derivados, os números de Salas de Internação e Salas de Cirurgia totais, sendo que, sempre que for aberta uma nova sala em um hospital, uma desses atributos tem seu valor acrescido em um. Além disso, todo hospital pode ter também estar associado a um ou vários tipos de Autorização SNT. Quando um hospital recebe um tipo de Autorização SNT, também é atribuída uma validade para o relacionamento entre o Hospital e a Autorização.

No interior dos hospitais existem entidades associadas que representam recursos físicos indispensáveis. Os Laboratórios, modelados como entidades fracas, respondem pela execução de exames essenciais, como histocompatibilidade e tipagem sanguínea. Cada laboratório pode realizar vários Tipos de Exame, e cada Exame relaciona-se necessariamente com apenas um Tipo de Exame. As Salas, também entidades fracas, podem ser de internação ou cirúrgicas, e incluem atributos como número, tipo e tempo necessário de higienização entre usos. Esse tempo de higienização deve ser respeitado entre procedimentos. Uma Sala de Internação tem como atributo o seu número de leitos, enquanto que uma Sala Cirúrgica necessariamente se relaciona com os Equipamentos que tem, sendo especificada a quantidade que a sala tem de cada tipo de equipamento. Cada tipo de equipamento também poderá ter uma descrição explicando seu uso.

A entidade Tipo de Órgão desempenha um papel específico, pois registra o nome do orgão em questão. Essa abstração é necessária porque receptores não aguardam um órgão individual, mas sim um tipo, cuja disponibilidade precisa ser acompanhada pelo sistema. Por esse motivo, o tipo de órgão se relaciona à Cirurgia de Coleta (de forma que a cirurgia necessariamente se relaciona com um ou vários Tipos) e aos pacientes doadores e receptores. A agregação Órgão Coletado, formado pela relação entre Tipo de Orgão e Cirurgia de Coleta, possui chave composta formada pelas chaves da Cirurgia de Coleta e do Tipo de Orgão junto com o Lado do orgão. O Lado do orgão usado como parte da chave é, por exemplo, a informação de se é um pulmão direito ou esquerdo, ou, caso o orgão seja único, como no caso de um coração, armazena-se ”Indiferente”como o Lado.

As Cirurgias são outro processo central, sendo formadas por uma agregação no relacionamento entre Paciente e Sala Cirúrgica, e tendo como chave composta a chave da Sala Cirúrgica, do Paciente e a sua data. Sua identificação única resulta da combinação entre o nome do paciente, o número da sala

de cirurgia, o hospital onde foi realizada e a data da operação. Toda cirurgia exige médicos responsáveis pela operação e enfermeiros para auxiliar, podendo ser classificada como de coleta ou de recepção (ou como nenhuma das duas, caso a classificação não se aplique, como em uma cirurgia comum). A Cirurgia de Recepção necessariamente se relaciona com um ou vários orgãos coletados, e um orgão coletado se relaciona a apenas uma Cirurgia de Recepção.

A Internação é formada por uma agregação no relacionamento entre a Sala de Internação e um Paciente, e tem como chave do Paciente, da Sala e a data de entrada da internação. Ela controla a permanência do paciente em leitos hospitalares. E também armazenada como atributo a data esperada de término da internação. Já os Exames são uma agregação formada no relacionamento entre Laboratório e Paciente, tomando como chave a chave desses dois junto com a data e horário do exame. Como atributo, também armazena-se o resultado do exame. Também, como dito antes, um Exame necessariamente se relaciona com uma única entidade Tipo de Exame, e esse Tipo de Exame deve ser um que o Laboratório associado consiga realizar.

1.3. Restrições de integridade

- Os tipos de cirurgia recepção e coleta só podem ser criadas com pacientes receptores e doadores, respectivamente.
- Os atributos CRM e COREN para médicos e enfermeiros, respectivamente, são únicos.
- Um médico ou enfermeiro não pode atuar enquanto estiver em cirurgia ou internado
- O médico responsável não pode ser também o paciente do exame.
- Salas devem respeitar o tempo de higienização entre procedimentos.
- Um mesmo paciente pode acumular categorias, mas não pode atuar como doador e receptor para o mesmo tipo de órgão
- Após suas respectivas cirurgias, doador e receptor devem perder seus relacionamentos de “Doa” e “Espera” para o tipo de órgão relacionado
- A ordem temporal deve ser respeitada: coleta $\rightarrow$ recepção
- Todo Exame deve ser de um tipo que o seu respectivo laboratório realiza
- O lado de um órgão pode ser ”Esquerdo”, ”Direito”ou ”Indiferente”, dependendo do tipo de órgão e do lado onde foi coletado.

1.4. Principais Operações

O sistema permitirá:

- Cadastros: inserção de novos pacientes, profissionais, hospitais e autorizações SNT.
- Atualizações: edição de dados clínicos, registros hospitalares e disponibilidade de salas.
- Consultas:
  - Doadores compatíveis para um receptor específico.
  - Histórico de transplantes por hospital.
  - Equipes médicas associadas a determinada cirurgia.
  - Pacientes internados em um período.
  - Autorizações SNT em situação de vencimento.

1.5. Funcionalidades Principais

- Cadastro e gerenciamento de pessoas: pacientes, doadores, receptores, médicos e enfermeiros.
- Cadastro e atualização de hospitais: incluindo suas autorizações SNT.
- Gerenciamento de laboratórios e exames: múltiplos exames por paciente possíveis.
- Gerenciamento de salas hospitalares: controle de recursos de internação e cirurgia.
- Registro de processos de transplante: coleta, transporte, recepção, cirurgia e internação.
- Consultas complexas: histórico de transplantes, disponibilidade de recursos, compatibilidade entre doador e receptor.

2. Modelo Entidade-Relacionamento (MER)

2.1. Diagrama MER

O diagrama elaborado contempla:

- Entidade Pessoa com especializações: Funcionário (Médico - com especialização - / Enfermeiro) e Paciente (Doador / Receptor).
- Entidade Hospital, com laboratórios (entidade fraca) e salas (entidade fraca), com tempo de higienização controlado, e podem possuir Autorizações SNT.
- Cirurgia, vinculada a médicos, enfermeiros e sala utilizada.
- Recepção, registrando chegada de órgãos e associação ao paciente e sala cirúrgica.
- Coleta, representando a retirada de órgãos do doador no hospital de origem.
- Internação, acompanhando a permanência do paciente em leito hospitalar.
- Exame, realizado em laboratório com resultados associados, sendo que um laboratório pode realizar vários Tipos de Exame.
- Órgão coletado, representa o órgão coletado pela coleta e recebido pela recepção
- Relacionamentos que garantem rastreabilidade de ponta a ponta no processo de transplantes.

2.2. Análise de ciclos

- **Ciclo 1: Funcionário → Hospital → Sala → Internação → Paciente**

  Um funcionário do hospital pode também ser um paciente. A aplicação deve identificar quando ele estiver em cirurgia/internado e impedir que o funcionário atue nos seus serviços enquanto estiver internado.

- **Ciclo 2: Paciente → Cirurgia → Médico/Enfermeiro**

  Assim como no caso anterior, um funcionário do hospital pode também ser um paciente. A aplicação deve identificar quando ele estiver em cirurgia/internado e impedir que o funcionário atue nos seus serviços enquanto estiver internado.

- **Ciclo 3: Paciente → Exame → Médico**

  Assim como nos casos anteriores, um médico pode ser também um paciente. Desta forma, a aplicação deve impedir que o médico supervisor do exame seja também o paciente do exame.

- **Ciclo 4: Coleta → Órgão Coletado → Recepção**

  Como coleta e recepção são especializações de cirurgia, cria-se um ciclo entre elas. O paciente da cirurgia de recepção não pode receber o mesmo órgão removido na cirurgia de coleta.

- **Ciclo 5: Doador → Tipo de órgão → Receptor**

  Como doador e receptor são especializações de paciente, cria-se um ciclo entre eles. A regra de restrição “Um mesmo paciente pode acumular categorias, mas não pode atuar como doador e receptor para o mesmo órgão” resolve esse ciclo, impedindo possíveis problemas.

- **Ciclo 6: Laboratório → Tipo de Exame → Exame**

  Como todo laboratório tem um conjunto de tipos de exame que ele realiza, todos os exames feitos pelo mesmo devem ser de algum dos tipos relacionados ao laboratório.

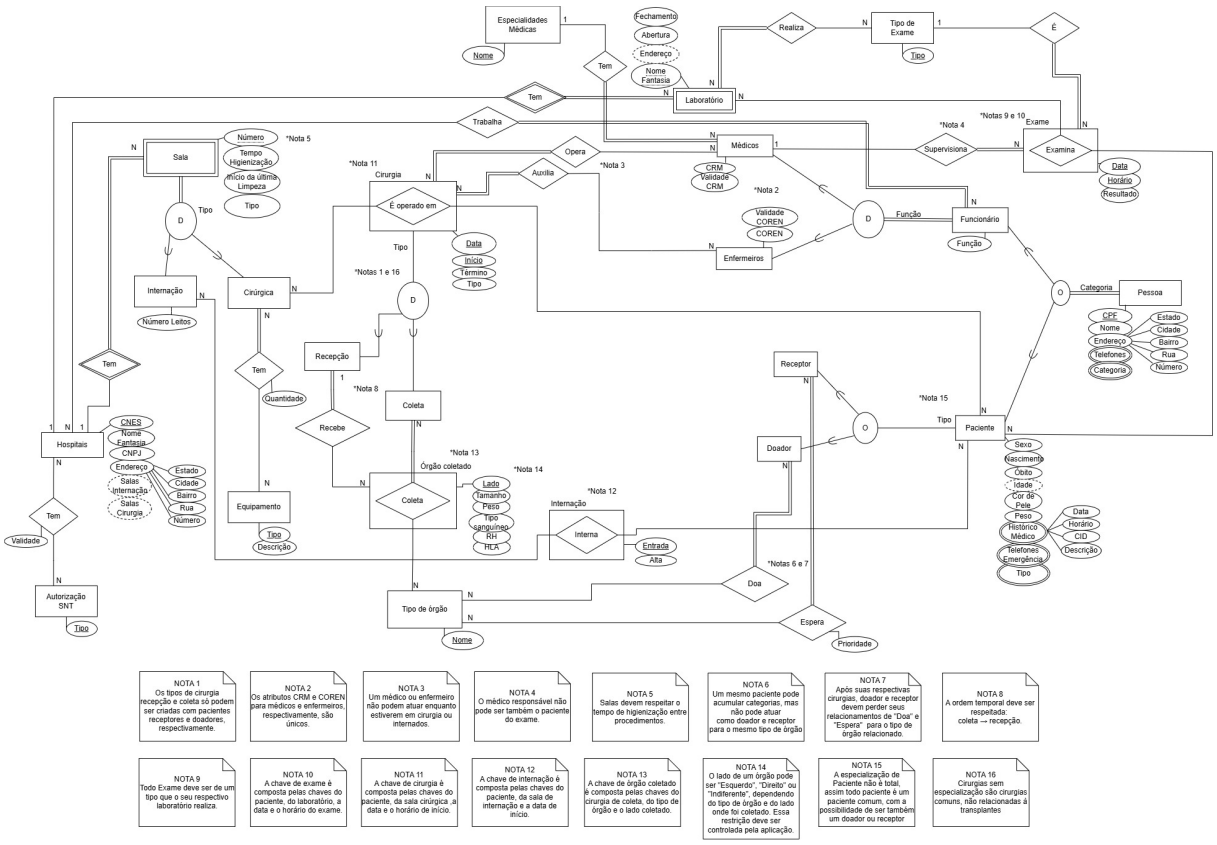


Figura 1: *Diagrama Entidade-Relacionamento.*

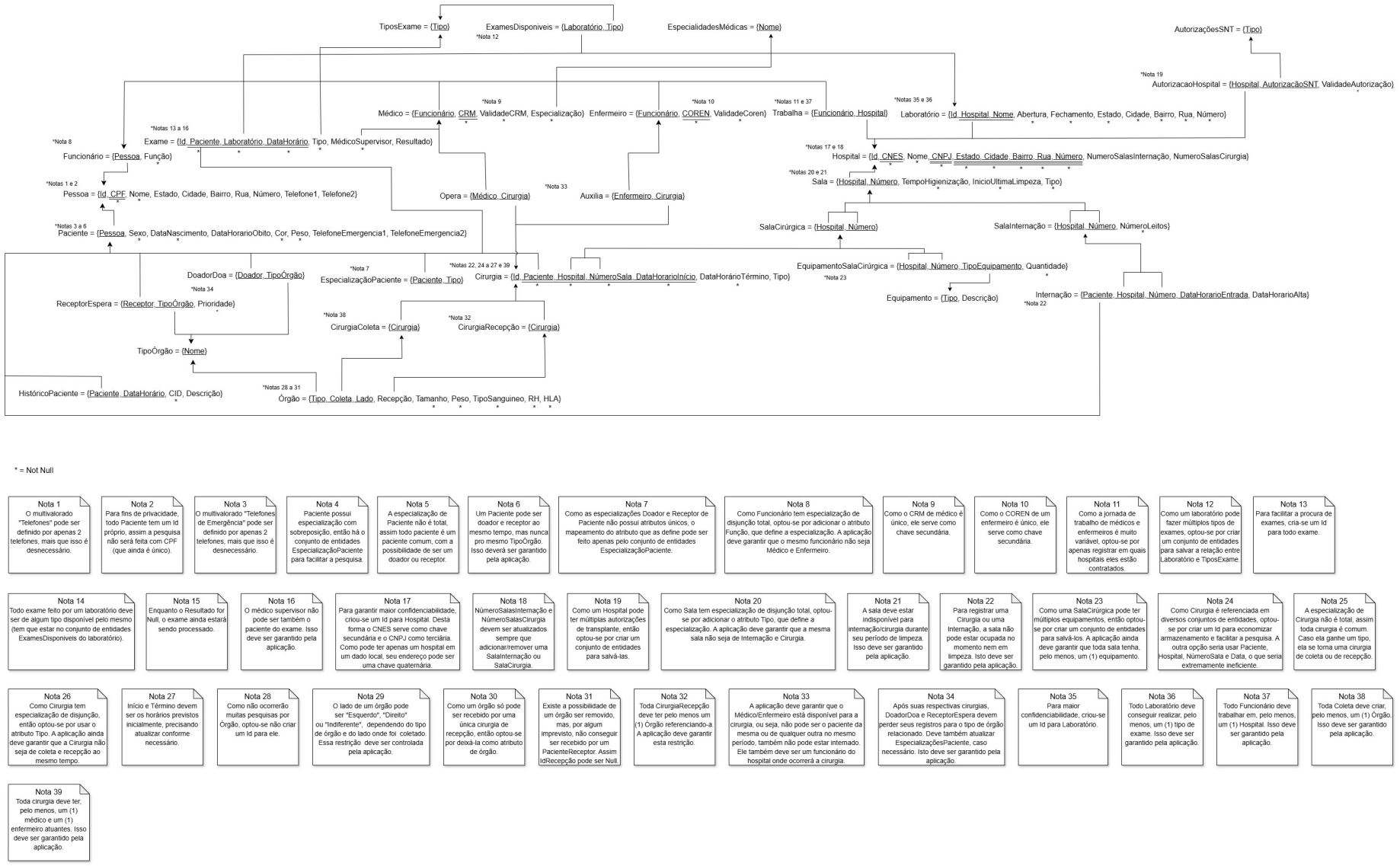
2.3. Mudanças com relação à última entrega

- **Hospital:** Atributos relacionados a número de vagas substituído por números de salas totais, e os atributos relacionados a veículos foram removidos.
- **Veículos:** A entidade Veículos e seus relacionamentos foram completamente removidos, por decisão de escopo de projeto.
- **Autorização:** Deixou de ser um atributo de “Hospital” e passou a ser uma entidade denominada “Autorização SNT”, para fins de padronização.
- **Equipamento:** Deixou de ser um atributo de “Sala Cirúrgica” e passou a ser uma entidade denominada “Equipamento”, que se relaciona com a sala, também para fins de padronização.
- **Especializações (Cirurgia e Paciente):** As especializações de tipo de cirurgia e de paciente foram mudadas para parciais, para que o banco de dados armazene cirurgias que não são transplantes e pacientes que não são nem doadores, nem receptores.
- **Agregação “Órgão Coletado”:** Uma entidade “Orgão Coletado” foi criada à partir da agregação do relacionamento Coleta entre “Cirurgia de Coleta” e “Tipo de Orgão”.
- **Atributo “Prioridade” (Receptor):** Deixou de ser atributo de “Receptor” e passou a ser atributo do relacionamento “Espera”, permitindo que ele tenha diferentes prioridades para cada tipo de órgão.
- **Tipo de Exame:** O atributo “Tipo” da agregação “Exame” passou a ser uma entidade “Tipo de Exame”, novamente para fins de padronização.
- **Médico e Enfermeiro:** Chaves primárias removidas, pois herdam a chave primária de “Pessoa”.
- **Tipo de Órgão:** Chave primária “Id” removida, agora sendo “Nome”.
- **Notas:** Adicionadas notas explicativas que faltavam no diagrama.

3. Modelo Relacional

Neste tópico, será introduzido o modelo relacional do banco de dados proposto. Nele, é estabelecida a ligação entre as tabelas físicas. Além de apresentar o esquema, serão examinados também os mapeamentos feitos, observando suas motivações.

No diagrama (Figura 2), as tabelas são representadas por blocos e relacionadas utilizado setas.



* = Not Null

Nota 1
O multivalorado "Telefones" pode ser definido por apenas 2 telefones, mais que isso é desnecessário.

Nota 2
Para fins de privacidade, todo Paciente tem um Id próprio, assim a pesquisa não será feita com CPF (que ainda é único).

Nota 3
O multivalorado "Telefones de Emergência" pode ser definido por apenas 2 telefones, mais que isso é desnecessário.

Nota 4
Paciente possui especialização com sobreposição, então há o conjunto de entidades EspecializaçãoPaciente para facilitar a pesquisa.

Nota 5
A especialização de Paciente não é total, assim todo paciente é um paciente comum, com a possibilidade de ser um doador ou receptor.

Nota 6
Um Paciente pode ser doador e receptor ao mesmo tempo, mas nunca pro mesmo TipoÓrgão. Isso deverá ser garantido pela aplicação.

Nota 7
Como as especializações Doador e Receptor de Paciente não possui atributos únicos, o mapeamento do atributo que as define pode ser feito apenas pelo conjunto de entidades EspecializaçãoPaciente.

Nota 8
Como Funcionário tem especialização de disjunção total, optou-se por adicionar o atributo Função, que define a especialização. A aplicação deve garantir que o mesmo funcionário não seja Médico e Enfermeiro.

Nota 9
Como o CRM de médico é único, ele serve como chave secundária.

Nota 10
Como o COREN de um enfermeiro é único, ele serve como chave secundária.

Nota 11
Como a jornada de trabalho de médicos e enfermeiros é muito variável, optou-se por apenas registrar em quais hospitais eles estão contratados.

Nota 12
Como um laboratório pode fazer múltiplos tipos de exames, optou-se por criar um conjunto de entidades para salvar a relação entre Laboratório e TiposExame.

Nota 13
Para facilitar a procura de exames, cria-se um Id para todo exame.

Nota 14
Todo exame feito por um laboratório deve ser de algum tipo disponível pelo mesmo (tem que estar no conjunto de entidades ExamesDisponiveis do laboratório).

Nota 15
Enquanto o Resultado for Null, o exame ainda estará sendo processado.

Nota 16
O médico supervisor não pode ser também o paciente do exame. Isso deve ser garantido pela aplicação.

Nota 17
Para garantir maior confidenciabilidade, criou-se um Id para Hospital. Desta forma o CNES serve como chave secundária e o CNPJ como terciária. Como pode ter apenas um hospital em um dado local, seu endereço pode ser uma chave quaternária.

Nota 18
NúmeroSalasInternação e NúmeroSalasCirurgia devem ser atualizados sempre que adicionar/remover uma SalaInternação ou SalaCirurgia.

Nota 19
Como um Hospital pode ter múltiplas autorizações de transplante, então optou-se por criar um conjunto de entidades para salvá-las.

Nota 20
Como Sala tem especialização de disjunção total, optou-se por adicionar o atributo Tipo, que define a especialização. A aplicação deve garantir que a mesma sala não seja de Internação e Cirurgia.

Nota 21
A sala deve estar indisponível para internação/cirurgia durante seu período de limpeza. Isso deve ser garantido pela aplicação.

Nota 22
Para registrar uma Cirurgia ou uma Internação, a sala não pode estar ocupada no momento nem em limpeza. Isto deve ser garantido pela aplicação.

Nota 23
Como uma SalaCirúrgica pode ter múltiplos equipamentos, então optou-se por criar um conjunto de entidades para salvá-los. A aplicação ainda deve garantir que toda sala tenha, pelo menos, um (1) equipamento.

Nota 24
Como Cirurgia é referenciada em diversos conjuntos de entidades, optou-se por criar um Id para economizar armazenamento e facilitar a pesquisa. A outra opção seria usar Paciente, Hospital, NúmeroSala e Data, o que seria extremamente ineficiente.

Nota 25
A especialização de Cirurgia não é total, assim toda cirurgia é comum. Caso ela ganhe um tipo, ela se torna uma cirurgia de coleta ou de recepção.

Nota 26
Como Cirurgia tem especialização de disjunção, então optou-se por usar o atributo Tipo. A aplicação ainda deve garantir que a Cirurgia não seja de coleta e recepção ao mesmo tempo.

Nota 27
Início e Término devem ser os horários previstos inicialmente, precisando atualizar conforme necessário.

Nota 28
Como não ocorrerão muitas pesquisas por Órgão, optou-se não criar um Id para ele.

Nota 29
O lado de um órgão pode ser "Esquerdo", "Direito" ou "Indiferente", dependendo do tipo de órgão e do lado onde foi coletado. Essa restrição deve ser controlada pela aplicação.

Nota 30
Como um órgão só pode ser recebido por uma única cirurgia de recepção, então optou-se por deixá-la como atributo de órgão.

Nota 31
Existe a possibilidade de um órgão ser removido, mas, por algum imprevisto, não conseguir ser recebido por um PacienteReceptor. Assim, IdRecepção pode ser Null.

Nota 32
Toda CirurgiaRecepção deve ter pelo menos um (1) Órgão referenciando-a. A aplicação deve garantir esta restrição.

Nota 33
A aplicação deve garantir que o Médico/Enfermeiro está disponível para a cirurgia, ou seja, não pode ser o paciente da mesma ou de qualquer outra no mesmo período, também não pode estar internado. Ele também deve ser um funcionário do hospital onde ocorrerá a cirurgia.

Nota 34
Após suas respectivas cirurgias, DoadorDoa e ReceptorEspera devem perder seus registros para o tipo de órgão relacionado. Deve também atualizar EspecializaçõesPaciente, caso necessário. Isto deve ser garantido pela aplicação.

Nota 35
Para maior confidenciabilidade, criou-se um Id para Laboratório.

Nota 36
Todo Laboratório deve conseguir realizar, pelo menos, um (1) tipo de exame. Isso deve ser garantido pela aplicação.

Nota 37
Todo Funcionário deve trabalhar em, pelo menos, um (1) Hospital. Isso deve ser garantido pela aplicação.

Nota 38
Toda Coleta deve criar, pelo menos, um (1) Órgão. Isso deve ser garantido pela aplicação.

Nota 39
Toda cirurgia deve ter, pelo menos, um (1) médico e um (1) enfermeiro atuantes. Isso deve ser garantido pela aplicação.

Figura 2: *Esquema relacional.*

3.1. Discussão sobre mapeamentos

1. Especialização de Pessoa

Solução adotada: Foi mapeada na tabela “Pessoa” os dados gerais de cada pessoa, criando, então, uma tabela para suas especializações “Funcionário” e “Paciente”.

Vantagens: Separando funcionários de pacientes, os relacionamentos ficam mais consistentes e confiáveis, uma vez que só poderão ser feitos com pessoas da categoria correta.

Desvantagens: A repetição da chave primária de “Pessoa” acaba por ocupar mais espaço de armazenamento. A adição de pessoas também acaba tendo um custo de processamento mais elevado, uma vez que devem ser adicionadas em múltiplas tabelas, fora a necessidade da aplicação de garantir a especialização total.

Soluções alternativas: Todas as informações poderiam ser salvas diretamente em “Pessoa”, por meio da adição de múltiplos atributos para suas especializações. Em troca isso iria requerer múltiplas análises por parte da aplicação em toda interação com a base de dados, aumentando consideravlemente o custo computacional. Poderíamos também incluir uma tabela “EspecializaçãoPessoa”, mas ela não teria grande utilidade, uma vez que teria mais tuplas que as demais, perdendo a vantagem na pesquisa, e aumentaria o espaço de armazenamento e custo computacional para novas inserções.

2. Especialização de Funcionário

Solução adotada: Na tabela “Funcionário”, existe apenas a chave primária de identificação de “Pessoa” e seu atributo de especialização “Função”. Como todo funcionário deve ter uma função, este atributo não pode ser nulo. Suas especializações, “Médico” e “Enfermeiro”, possuem suas próprias tabelas.

Vantagens: Separando os tipos de funcionário, os relacionamentos ficam mais consistentes e confiáveis, uma vez que só poderão ser feitos com funcionários da função correta. O atributo de função auxilia em garantir a disjunção e totalidade da especialização, além de facilitar a pesquisa.

Desvantagens: A repetição da chave primária de ”Funcionário”acaba por aumentar o custo de armazenamento. Apesar de auxiliar, o uso de “Função” não garante a especialização total, restando à aplicação garantir que a especialização será criada junto à criação do funcionário, além de verificar que está sendo inserido na especialização correta e apenas nela, para garantir a disjunção.

Soluções alternativas: Todas as informações poderiam ser salvas diretamente em “Funcionário”, por meio da adição de múltiplos atributos para suas especializações. Em troca isso iria requerer múltiplas análises por parte da aplicação em toda interação com a base de dados, aumentando consideravlemente o custo computacional.

3. Especialização de Paciente

Solução adotada: Como as especializações “Doador” e “Receptor” de “Paciente” não possuem atributos únicos e não serão referenciadas muitas vezes, o mapeamento delas é feito apenas pela tabela “EspecializaçãoPaciente”, que armazena quais são as especializações dos pacientes, se eles tiverem uma.

Vantagens: A tabela “EspecializaçãoPaciente” permite, de forma simples, definir a especialização de sobreposição. Como haverão mais pacientes sem relação com transplante de órgãos do que aqueles relacionados, o mapeamento da especialização estar numa tabela à parte acaba também por poupar espaço de armazenamento e diminuir o tempo de pesquisa pelas especializações do paciente.

Desvantagens: Como não existem tabelas específicas para cada especialização, os relacionamentos com “Doador” e “Receptor” devem ser verificados pela aplicação para garantir que são pacientes do tipo correto. Quando um paciente ganhar uma especialização, a aplicação deverá garantir que ele ganhará um relacionamento correspondente com “TipoÓrgão” para garantir a participação total. Quando o paciente passar por uma cirurgia, será necessário verificar se a especialização ainda é válida, criando mais necessidade de processamento.

Soluções alternativas: Todas as informações poderiam ser salvas diretamente em “Paciente”, uma vez que eles não possuem atributos únicos. No entanto, isso criaria um aumento no custo de armazenamento, uma vez que há muito mais pacientes sem relação com transplantes de órgãos, o que também aumentaria o tempo de pesquisa pelas especializações do paciente.

4. Especialização de Cirurgia

Solução adotada: Na tabela “Cirurgia”, existem os dados gerais da cirurgia, além de seu atributo de especialização “Tipo”. Como nem toda cirurgia terá relação com transplantes, “Tipo” pode ser nulo. Suas especializações “CirurgiaColeta” e “CirurgiaRecepção” não possuem atributos específicos, mas podem fazer múltiplos relacionamentos com outras tabelas, assim se tornando relevante separá-las.

Vantagens: Separando os tipos de cirurgia, os relacionamentos ficam mais consistentes e confiáveis, uma vez que só poderão ser feitos com cirurgias de tipos corretos. O atributo de tipo auxilia em garantir a disjunção da especialização, além de facilitar a pesquisa.

Desvantagens: A repetição da chave primária de “Cirurgia” acaba por aumentar o custo de armazenamento, principalmente pelo fato de que suas especializações não possuírem atributos únicos, apenas a referência. Apesar de auxiliar, o uso de “Tipo” não garante a disjunção, restando à aplicação garantir que a especialização será inserida na tabela correta.

Soluções alternativas: Poderia remover as tabelas das especializações, uma vez que não trazem informações adicionais. No entanto isso aumentaria o custo de inserção de novos órgãos, uma vez que seria necessária fazer uma pesquisa pela tabela “Cirurgia”, que é muito maior, além de ser necessária uma análise do tipo.

5. Especialização de Sala

Solução adotada: Na tabela “Sala”, existem os dados gerais da sala, além de seu atributo de especialização “Tipo”. Como toda sala tem uma função específica, “Tipo” não pode ser nulo. Suas especializações “SalaCirúrgica” e “SalaInternação” não possuem muitos atributos específicos, mas podem fazer múltiplos relacionamentos com outras tabelas, assim se tornando relevante separá-las.

Vantagens: Separando os tipos de sala, os relacionamentos ficam mais consistentes e confiáveis, uma vez que só poderão ser feitos com salas de tipos corretos. O atributo de tipo não nulo auxilia em garantir a disjunção e a totalidade da especialização, além de facilitar a pesquisa.

Desvantagens: A repetição da chave primária de “Sala” acaba por aumentar o custo de armazenamento, principalmente pelo fato de suas especializações não possuírem muitos atributos únicos. Apesar de auxiliar, o uso de “Tipo” não garante a disjunção, restando à aplicação garantir que a especialização será inserida na tabela correta, nem sua totalidade, novamente restando à aplicação garantir que a inserção ocorrerá também na sua tabela de especialização definida.

Soluções alternativas: Todas as informações poderiam ser salvas diretamente em “Sala”, por meio da adição do atributo “NúmeroLeitos”, o único atributo único de suas especializações. Em troca isso iria requerer múltiplas análises por parte da aplicação em toda interação com a base de dados, aumentando consideravlemente o custo computacional.

6. **Relacionamento “É” entre “Tipo de Exame” e “Exame” - 1:N**

Solução adotada: Como esse relacionamento serve para definir qual o tipo de exame do exame, simplesmente adicionou-se uma chave estrangeira que refere a um valor de “TiposExame”, a qual não pode ser nula.

Vantagens: Garante a totalidade e a unicidade do relacionamento, onde todo exame pode ser de um único tipo.

Desvantagens: Não foram identificadas desvantagens em mapear desta maneira.

Soluções alternativas: Poderia ser feita uma tabela relacionando os exames ao seu tipo, onde exame seria a única chave primária. Isso aumentaria o custo de armazenamento, pois teria a repetição da chave primária de “Exame”, além de aumentar o custo de processamento, pois seriam necessárias 2 pesquisas para coletar todas as informações do exame.

7. **Relacionamento “Tem” entre “Médico” e “Especialidades Médicas” - 1:N**

Solução adotada: Como esse relacionamento serve para definir qual a especialização do médico, optou-se por adicioná-lo como uma chave estrangeira em “Médico” que refere a um valor de “Especialidades Médicas”. Ele sendo não-nulo garante também sua participação total.

Vantagens: Garante a totalidade e a unicidade do relacionamento, onde todo médico pode ter apenas uma especialização.

Desvantagens: Não foram identificadas desvantagens em mapear desta maneira.

Soluções alternativas: Poderia ser feita uma tabela relacionando os médicos às especializações, onde médico seria a única chave primária. Isso aumentaria o custo de armazenamento, pois teria a repetição da chave primária de “Médico”, além de aumentar o custo de processamento, pois seriam necessárias 2 pesquisas para coletar todas as informações do médico.

8. **Relacionamento “Supervisiona” entre “Médico” e “Exame” - 1:N**

Solução adotada: Como esse relacionamento serve para definir qual o médico responsável pelo exame, simplesmente adicionou-se uma chave estrangeira que refere a um valor de “Médico”, a qual não pode ser nula.

Vantagens: Garante a totalidade e a unicidade do relacionamento, onde todo exame pode ser supervisionado por um único médico.

Desvantagens: Não foram identificadas desvantagens em mapear desta maneira.

Soluções alternativas: Poderia ser feita uma tabela relacionando os exames ao médico, onde exame seria a única chave primária. Isso aumentaria o custo de armazenamento, pois teria a repetição da chave primária de “Exame”, além de aumentar o custo de processamento, pois seriam necessárias 2 pesquisas para coletar todas as informações do exame.

9. **Relacionamento “Recebe” entre “Recepção” e “Órgão Coletado” - 1:N**

Solução adotada: Esse relacionamento serve para definir quais órgãos foram recebidos na cirurgia de recepção. Alternativamente, pode-se interpretar como qual cirurgia recebeu o órgão. Dessa forma, optou-se por adicionar uma chave estrangeira que refere a um valor de “CirurgiaRecepção”, a qual pode ser nula. Como consequência, se torna necessária a influência da aplicação para garantir que toda cirurgia de recepção seja referenciada por, pelo menos, um órgão.

Vantagens: Garante a unicidade do relacionamento, onde todo órgão pode ser recebido por uma única cirurgia de recepção.

Desvantagens: Não garante a totalidade da relação por parte da cirurgia, assim sendo necessária a influência da aplicação.

Soluções alternativas: Poderia ser feita uma tabela relacionando os órgãos às cirurgias, onde órgão seria a única chave primária. Isso aumentaria o custo de armazenamento, pois teria a repetição da chave primária de "Órgão", além de aumentar o custo de processamento, pois seriam necessárias 2 pesquisas para encontrar todas as informações dos órgãos.

10. Relacionamento fraco "Tem" entre "Hospital" e "Sala" - 1:N

Solução adotada: A forma padrão de lidar com entidades fracas, simplesmente cria uma chave composta usando a chave primária da entidade forte com a chave fraca da entidade fraca.

Vantagens: Garante unicidade das chaves.

Desvantagens: Não foram identificadas desvantagens em mapear desta maneira.

Soluções alternativas: Não foram identificadas alternativas convencionais para esse mapeamento.

11. Relacionamento fraco "Tem" entre "Hospital" e "Laboratório" - 1:N

Solução adotada: A forma padrão de lidar com entidades fracas, simplesmente cria uma chave composta usando a chave primária da entidade forte com a chave fraca da entidade fraca. Esta entidade conta com uma chave primária artificial, a qual é relevante para motivos de segurança que serão discutidos posteriormente. Desta forma a chave composta atua como secundária.

Vantagens: Garante unicidade das chaves.

Desvantagens: Não foram identificadas desvantagens em mapear desta maneira.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

12. Relacionamento "Tem" entre "Hospital" e "Autorizações SNT" - N:N

Solução adotada: Como o relacionamento é N:N, podendo variar consideravelmente em termos de quantidade, optou-se por criar uma tabela à parte com todos os valores do conjunto de relacionamentos.

Vantagens: Preserva a cardinalidade do relacionamento.

Desvantagens: Maior custo de armazenamento.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

13. Relacionamento "Trabalha" entre "Hospital" e "Funcionário" - N:N

Solução adotada: Como o relacionamento é N:N, podendo variar consideravelmente em termos de quantidade, optou-se por criar uma tabela à parte com todos os valores do conjunto de relacionamentos. Por conta da participação total do funcionário, será necessária a influência da aplicação.

Vantagens: Preserva a cardinalidade do relacionamento.

Desvantagens: Maior custo de armazenamento, necessita de manipulação por parte da aplicação.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

14. Relacionamento “Tem” entre “Sala Cirúrgica” e “Equipamento” - N:N

Solução adotada: Como o relacionamento é N:N, podendo variar consideravelmente em termos de quantidade, optou-se por criar uma tabela à parte com todos os valores do conjunto de relacionamentos. Por conta da participação total da sala, será necessária a influência da aplicação.

Vantagens: Preserva a cardinalidade do relacionamento.

Desvantagens: Maior custo de armazenamento, necessita de manipulação por parte da aplicação.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

15. Relacionamento “Doa” entre “Doador” e “Tipo de Órgão” - N:N

Solução adotada: Como o relacionamento é N:N, podendo variar consideravelmente em termos de quantidade, optou-se por criar uma tabela à parte com todos os valores do conjunto de relacionamentos. Por conta da participação total do doador, será necessária a influência da aplicação.

Vantagens: Preserva a cardinalidade do relacionamento.

Desvantagens: Maior custo de armazenamento, necessita de manipulação por parte da aplicação.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

16. Relacionamento “Espera” entre “Receptor” e “Tipo de Órgão” - N:N

Solução adotada: Como o relacionamento é N:N, podendo variar consideravelmente em termos de quantidade, optou-se por criar uma tabela à parte com todos os valores do conjunto de relacionamentos. Por conta da participação total do receptor, será necessária a influência da aplicação.

Vantagens: Preserva a cardinalidade do relacionamento.

Desvantagens: Maior custo de armazenamento, necessita de manipulação por parte da aplicação.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

17. Relacionamento “Opera” entre “Médico” e “Cirurgia” - N:N

Solução adotada: Como o relacionamento é N:N, podendo variar consideravelmente em termos de quantidade, optou-se por criar uma tabela à parte com todos os valores do conjunto de relacionamentos. A participação total da cirurgia deve ser garantida pela aplicação.

Vantagens: Preserva a cardinalidade do relacionamento.

Desvantagens: Maior custo de armazenamento. Necessidade de manipulação da aplicação.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

18. Relacionamento “Auxilia” entre “Enfermeiro” e “Cirurgia” - N:N

Solução adotada: Como o relacionamento é N:N, podendo variar consideravelmente em termos de quantidade, optou-se por criar uma tabela à parte com todos os valores do conjunto de relacionamentos. A participação total da cirurgia deve ser garantida pela aplicação.

Vantagens: Preserva a cardinalidade do relacionamento.

Desvantagens: Maior custo de armazenamento. Necessidade de manipulação da aplicação.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

19. Relacionamento “Realiza” entre “Tipo de Exame” e “Laboratório” - N:N

Solução adotada: Como o relacionamento é N:N, podendo variar consideravelmente em termos de quantidade, optou-se por criar uma tabela à parte com todos os valores do conjunto de relacionamentos. A participação total do laboratório deve ser garantida pela aplicação.

Vantagens: Preserva a cardinalidade do relacionamento.

Desvantagens: Maior custo de armazenamento. Necessidade de manipulação da aplicação.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

20. Relacionamento “Examina” entre “Laboratório” e “Paciente” - N:N

Solução adotada: Como existe uma agregação neste relacionamento, o mapeamento foi feito criando uma tabela “Exame”, com chave primária artificial, a qual é relevante para motivos de segurança que serão discutidos posteriormente. Ela possui uma chave composta, que serve como secundária, definida pelas chaves de “Paciente”, “Laboratório” e data e horário.

Vantagens: Preserva a cardinalidade do relacionamento, define bem a agregação.

Desvantagens: Maior custo de armazenamento.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

21. Relacionamento “Interna” entre “Sala Internação” e “Paciente” - N:N

Solução adotada: Como existe uma agregação neste relacionamento, o mapeamento foi feito criando uma tabela “Internação”. Ela possui uma chave primária composta definida pelas chaves de “Paciente”, “Sala Internação” e a data de entrada.

Vantagens: Preserva a cardinalidade do relacionamento, define bem a agregação.

Desvantagens: Maior custo de armazenamento.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

22. Relacionamento “Coleta” entre “Cirurgia de Coleta” e “Tipo de Órgão” - N:N

Solução adotada: Como existe uma agregação neste relacionamento, o mapeamento foi feito criando uma tabela “Órgão”. Ela possui uma chave primária composta definida pelas chaves de “TipoÓrgão”, “CirurgiaColeta” e o lado de onde o órgão foi removido. Esta ultima chave é relevante pois o corpo humano possui múltiplos órgãos iguais, mas nunca do mesmo lado do corpo. Desta forma ele pode ser definido pelo lado da remoção, caracterizada pelas opções: ”Esquerdo”, ”Direito”ou ”Indiferente”. O lado deve ser confirmado pela aplicação, para garantir que não seja escolhido um valor inválido para o tipo de órgão.

Vantagens: Preserva a cardinalidade do relacionamento, define bem a agregação.

Desvantagens: Maior custo de armazenamento, necessidade de manipulação da aplicação.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

23. Relacionamento “É operado em” entre “Sala Cirúrgica” e “Paciente” - N:N

Solução adotada: Como existe uma agregação neste relacionamento, o mapeamento foi feito criando uma tabela “Cirurgia”, com uma chave primária artificial, a qual é relevante para motivos que serão discutidos posteriormente. Ela possui uma chave composta, que serve como secundária, definida pelas chaves de “Paciente”, “SalaCirúrgica”, data e horário de início.

Vantagens: Preserva a cardinalidade do relacionamento, define bem a agregação.

Desvantagens: Maior custo de armazenamento.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento.

24. Atributo Multivalorado “Histórico Médico”

Solução adotada: Este atributo multivalorado não pode ser definido por um número arbitrário de atributos dentro de “Paciente”, desta forma optou-se por criar uma tabela “HistóricoPaciente”, com chave primária composta definida pela chave de “Paciente” e a data e horário. Ela também armazena o CID do evento, o qual não pode ser nulo, e uma descrição mais detalhada, a qual pode ser nula.

Vantagens: Permite a propriedade multivalorada sem limites de registros.

Desvantagens: Maior custo de armazenamento.

Soluções alternativas: Não foram identificadas alternativas para esse mapeamento, dado o número de registros impossível de ser definido.

25. Atributo Multivalorado “Tipo” (Paciente)

Solução adotada: Optou-se por armazenar este atributo multivalorado na tabela “EspecializaçãoPaciente”, utilizando as chaves primária de “Paciente”, além do seu tipo.

Vantagens: Permite a propriedade multivalorada e a pesquisa se torna mais eficiente, pois tem menos pacientes doadores e receptores do que pacientes em geral.

Desvantagens: Nenhuma desvantagem foi encontrada em relação à sua solução alternativa.

Soluções alternativas: Poderiam ser adicionados dois atributos, um para cada especialização, que podem ser nulos. Isso, no entanto, levaria a um aumento no custo de armazenamento, pois teriam muitos pacientes com especializações nulas, e no desempenho da pesquisa por pacientes doadores/receptores, pois há muito mais pacientes que não têm relação aos transplantes.

26. Atributo Multivalorado “Categoria” (Pessoa)

Solução adotada: Optou-se por não armazenar este atributo, uma vez que foi decidido que ele não traria muita utilidade para a base de dados. São raríssimas as situações onde é relevante saber as categorias de uma pessoa, sendo muito mais comum verificar se ela é uma categoria em específico, situação onde seria muito mais eficiente pesquisar pela pessoa na tabela da especialização correspondente.

Vantagens: Economiza espaço de armazenamento, permite a propriedade multivalorada.

Desvantagens: Na situação raríssima onde deseja-se saber as categorias de uma pessoa, serão necessárias duas pesquisas, aumentando o custo computacional.

Soluções alternativas: Poderiam ser adicionados dois atributos, um para cada especialização, que podem ser nulos. Isso aumentaria a eficiência da rara pesquisa pelas funções de uma pessoa, pois há menos pessoas do que a combinação de funcionários com pacientes, no entanto levaria a um aumento no custo de armazenamento, pois teriam muitas pessoas com especializações nulas, principalmente na categoria de funcionário. Outra opção seria criar uma tabela como a “EspecializaçãoPaciente”, que armazena as categorias de cada pessoa. No entanto isso aumentaria o custo de armazenamento, pois é uma tabela nova, e ainda teria uma pesquisa mais rápida apenas na situação especial comentada.

27. Atributos Multivalorados “Telefones Emergência” (Paciente) e “Telefones” (Pessoa)

Solução adotada: Optou-se por armazenar estes atributos como dois atributos simples nas suas respectivas tabelas, uma vez que não é necessário um número muito maior do que este de registros pode paciente ou pessoa.

Vantagens: Menor custo de armazenamento.

Desvantagens: Limitação da propriedade multivalorada.

Soluções alternativas: Poderiam ser criadas uma tabela para cada propriedade, assim como foi feito com os demais atributos multivalorados definidos anteriormente. No entanto isso é desnecessário, criando a necessidade de duas pesquisas para obter todas as informações do paciente.

28. Id artificial de “Pessoa”

Solução adotada: Para garantir confidenciabilidade dos dados das pessoas, criou-se um Id artificial, o qual só torna possível saber quem é por meio de permissões.

Vantagens: Maior confidenciabilidade dos dados.

Desvantagens: A adição de um identificador sintético acaba diminuindo o significado semântico da chave primária, que deixa de ser algo real, o CPF, para algo sintético. Fora isso, também aumenta a complexidade das pesquisas por CPF, pois deixou de ser a chave primária.

Soluções alternativas: Não foram identificadas alternativas para este mapeamento sem necessitar um aumento considerável de complexidade, como criptografia.

29. Id artificial de “Hospital”

Solução adotada: Para garantir confidenciabilidade das operações, contratações, internações, cirurgias, etc, criou-se um Id artificial, o qual só torna possível saber qual é o hospital por meio de permissões.

Vantagens: Maior confidenciabilidade dos dados.

Desvantagens: A adição de um identificador sintético acaba diminuindo o significado semântico da chave primária, que deixa de ser algo real, o CNES do hospital, para algo sintético. Fora isso, também aumenta a complexidade das pesquisas por CNES, pois deixou de ser a chave primária.

Soluções alternativas: Não foram identificadas alternativas para este mapeamento sem necessitar um aumento considerável de complexidade, como criptografia.

30. Id artificial de “Laboratório”

Solução adotada: Para garantir confidenciabilidade da origem dos exames, além de facilitar a pesquisa pelos laboratórios, criou-se um Id artificial, o qual só torna possível saber da onde veio o exame por meio de permissões.

Vantagens: Maior confidenciabilidade dos dados, pesquisa mais rápida. Como múltiplos exames podem referenciar o mesmo laboratório, a chave estrangeira ser apenas o Id acaba por se tornar também uma redução no custo de armazenamento.

Desvantagens: A adição de um identificador sintético acaba diminuindo o significado semântico da chave primária, que deixa de ser algo real, como o nome do laboratório, para algo sintético. Fora isso, também aumenta a complexidade das pesquisas pelos nomes, seja do laboratório ou do hospital, pois deixaram de ser a chave primária.

Soluções alternativas: Não foram identificadas alternativas para este mapeamento sem necessitar um aumento considerável de complexidade, como criptografia.

31. Id artificial de “Exame”

Solução adotada: Para facilitar a pesquisa dos exames, optou-se por criar uma chave artificial, substituindo sua enorme chave composta. Isto também adiciona uma camada de confidenciabilidade, uma vez que só se pode saber os detalhes de origem do exame por meio de permissões.

Vantagens: Pesquisa mais rápida, maior confidenciabilidade.

Desvantagens: A adição de um identificador sintético acaba diminuindo o significado semântico da chave primária, que deixa de ser algo real, como o paciente relacionado, para algo sintético. Fora isso, também aumenta a complexidade das pesquisas pelos dados de origem, como o paciente, pois deixaram de ser as chaves primárias.

Soluções alternativas: Poderia utilizar apenas a chave composta já definida, mas levaria à perda de todas as vantagens.

32. Id artificial de “Cirurgia”

Solução adotada: Para facilitar a pesquisa e referências das cirurgias, optou-se por criar uma chave artificial, substituindo sua enorme chave composta. Isto também adiciona uma camada de confidenciabilidade, uma vez que só se pode saber os detalhes da cirurgia por meio de permissões.

Vantagens: Pesquisa mais rápida. Como múltiplos órgãos podem referenciar a mesma cirurgia de coleta, além de poderem referenciarem uma cirurgia de recepção, a chave estrangeira ser apenas o Id acaba por se tornar também uma redução no custo de armazenamento. Uma última vantagem é a maior confidenciabilidade gerada.

Desvantagens: A adição de um identificador sintético acaba diminuindo o significado semântico da chave primária, que deixa de ser algo real, como o paciente que foi operado, para algo sintético. Fora isso, também aumenta a complexidade das pesquisas por dados de origem, como o paciente, pois deixaram de ser as chaves primárias.

Soluções alternativas: Poderia utilizar apenas a chave composta já definida, mas levaria à perda de todas as vantagens.

3.2. Mudanças com relação à última entrega

- **MER:**
  - **Especialização:** deixou de ser um atributo de “Médicos” e passou a ser um cojunto de relacionamentos com o conjunto de entidades denominado “Especialidades Médicas”, para fins de padronização.
- **Relacional:**
  - Correção da imagem: inclusão de “EspecializaçãoPessoa” (removido, explicação abaixo) e “HistóricoPaciente”, que estavam faltando.
  - “EspecializaçãoPessoa” foi removido, ficou perceptível que era não só desnecessário, mas prejudicial, trazendo mais desvantagens que vantagens.
  - “PacienteReceptor” e “PacienteDoador” foram substituídos por “ReceptorEspera” e “DoadorDoa”. Por consequência, mudaram-se as notas 7 e 34 para refletir as mudanças.
  - Em “HistóricoPaciente”, “Cirurgia” e “Exame” os atributos de data e horário foram transformados em atributos únicos que salvam ambos. Exemplo: Data, Início e Término de “Cirurgia” foram transformados em DataHorárioInício e DataHorárioTérmino.
  - “Abertura” e “Fechamento” de “Laboratório” agora são não nulos. Ficou perceptível que permitir que fossem nulos abria espaço para configurações sem sentido, tendo um horário de abertura mas não um de fechamento, ou vice-versa.

  - "UltimaLimpeza" de "Sala" se tornou "InicioUltimaLimpeza", para deixar explícito que é o horário que começou a limpeza. A coluna também se tornou não nula, para garantir consistência na base de dados.
  - "Obito" de "Paciente", "Entrada" e "Alta" de "Internação" viraram "DataHorarioObito", "DataHorarioEntrada" e "DataHorarioAlta". Essa mudança foi para deixar explícito que deve salvar não só a data, mas o horário também.
  - Adição de "EspecialidadesMédicas" para refletir as mudanças no MER.

- **"EspecializaçãoPessoa":** Percebeu-se que esta tabela era prejudicial, uma vez que não auxiliava na pesquisa, exceto no raro caso onde se deseja saber quais as categorias de uma "Pessoa", e aumentava o custo de armazenamento.
- **Especialização de "Paciente":** "PacienteReceptor" e "PacienteDoador" foram substituídos por "ReceptorEspera" e "DoadorDoa". Originalmente criados para servir como especialização, agora são os relacionamentos entre "Paciente" e enquoteTipoÓrgão. A especialização agora é definida simplesmente por "EspecializaçãoPessoa", pois as especializações não possuem atributos específicos.
- **"EspecialidadesMédicas"**: foi adicionada para padronizar a base de dados, criando uma tabela que deve ser referenciada pelos médicos, padronizando suas especializações.
- **Relacionamentos Espera e Doa:** agora são definidos por "ReceptorEspera" e "DoadorDoa".
- **Identificadores artificiais:** foram corrigidas as descrições das desvantagens dos identificadores artificiais.

4. Implementação

Nesta seção serão apresentados o protótipo de aplicação e a implementação da base de dados do sistema de apoio às Comissões Intra-Hospitalares de Doação de Órgãos e Tecidos para Transplante (CIHDOTT). O protótipo foi desenvolvido utilizando a linguagem Python 3, enquanto a base de dados foi implementada no SGBD Oracle.

Nesta fase do projeto foi estruturado o esquema completo da base de dados, incluindo todas as tabelas e suas respectivas restrições de integridade. Também foram elaborados os scripts de alimentação, responsáveis pela inserção das tuplas necessárias para a validação do modelo. Além disso, foram desenvolvidas cinco consultas de complexidade média e alta, que foram utilizadas tanto para testes quanto para demonstrar a capacidade de análise do sistema. Por fim, o grupo implementou um protótipo inicial integrado ao banco de dados, permitindo a execução das operações básicas de cadastro e consulta.

4.1. Requisitos do Sistema

O protótipo foi desenvolvido no Windows 10/11, com Python 3.13.6. As informações necessárias para a instalação da aplicação e das bibliotecas necessárias para executá-lo estão contidas no arquivo README.md, que podem ser encontrados no repositório: GitHub

Para a base de dados foi utilizada a IDE *SQL Developer - Release 24.3.1*. Todos os scripts SQL para montagem da estrutura, inserção dos dados básicos e selects podem também ser encontrados no repositório.

4.2. Busca e Inserção no Protótipo

Os comandos SQL utilizados na aplicação encontram-se integralmente no arquivo Aplicacao.py, responsável pela lógica de interação com o banco de dados Oracle. A seguir estão os trechos relevantes do código que implementam operações de inserção e consulta. Cada fragmento é apresentado com sua função correspondente e uma explicação sobre o propósito da operação no contexto do sistema.

Os primeiros usos de SQL aparecem na função *VerificaExistenciaPessoaPaciente*, quando a aplicação precisa verificar se uma pessoa já está cadastrada, seja na tabela PESSOA ou na tabela PACIENTE. O código responsável por essa etapa é:

```python
#Verifica a existência do CPF na base de dados, seja como Pessoa ou como Paciente
#0 == Nenhum Registro, retorna ID nulo
#1 == Registrado em Pessoa, retorna também o ID do registro
#2 == Registrado em Paciente (e, consequentemente, em Pessoa), retorna também o ID do registro
#-1 == Erro, retorna ID nulo
def VerificaExistenciaPessoaPaciente(pool, cpf):
    #Verifica se este CPF já está cadastrado como Pessoa
    #Como Pessoa tem especialização obrigatória, isso só acontecerá com Funcionários

    sqlSelectPessoa = "SELECT * FROM PESSOA WHERE CPF = :cpfPessoa"
    dados = {"cpfPessoa": cpf}

    try:
        #Pega uma conexão com o BD
        with pool.acquire() as conn:
            #Cria um cursor pra conexão
            with conn.cursor() as cursor:
                #cursor.execute trata os dados, protegendo contra injeções
                cursor.execute(sqlSelectPessoa, dados)
                rows = cursor.fetchall()

                #Como o CPF é único, só pode haver 0 ou 1 registro
                if len(rows) == 1:
                    #Pega o seu id de registro
                    idPessoaBytes = rows[0][0]

                    #Verifica se já está registrado como um Paciente
                    sqlSelectPaciente = "SELECT * FROM PACIENTE WHERE PESSOA = :idPessoa"
                    dados = {"idPessoa": idPessoaBytes}
                    #cursor.execute trata os dados, protegendo contra injeções
                    cursor.execute(sqlSelectPaciente, dados)
                    rows = cursor.fetchall()

                    #Novamente, ID é único, terá 0 ou 1 registro
                    if len(rows) == 1:
                        return 2, idPessoaBytes
                    #Se não estiver como Paciente
                    else:
                        return 1, idPessoaBytes
                else:
                    return 0, None
    except oracledb.Error as e:
        print(f"\nErro oracle: {e}\n")
        return -1, None
    except Exception as e:
        print(f"\nErro: {e}\n")
        return -1, None
```

Figura 3: *Selects de verificação de unicidade do CPF*

Esse trecho permite à aplicação distinguir três situações: pessoas sem cadastro, pessoas cadastradas que ainda não são pacientes e pessoas cadastradas que já possuem vínculo com a tabela PACIENTE. Sem essa verificação, a operação de inserção duplicada poderia violar a regra de integridade definida no esquema lógico: todo CPF é único.

Avançando para a função *InsertPessoaPaciente*, a aplicação executa os comandos de INSERT nas tabelas PESSOA e PACIENTE. O trecho é o seguinte:

```
#Após coletar os dados
try:
    #Pega uma conexão com o BD
    with pool.acquire() as conn:
        #Cria um cursor pra conexão
        with conn.cursor() as cursor:
            idPessoaBytes = dadosPessoa.get("ID_PESSOA_BYTES")

            #Se o paciente não estava pré-cadastrado como pessoa
            if idPessoaBytes is None:
                #Cria no cursor uma variável, necessário pra usar o RETURNING INTO
                idPessoaRet = cursor.var(oracledb.BINARY)

                #RETURNING INTO -> pega o ID criado no insert, que vai ser necessário pra próxima parte da inserção
                sqlInsertPessoa = \
                    "INSERT INTO PESSOA (CPF, NOME, ESTADO, CIDADE, BAIRRO, RUA, NUMERO, TELEFONE1, TELEFONE2) " \
                    "VALUES (:CPF, :NOME, :ESTADO, :CIDADE, :BAIRRO, :RUA, :NUMERO, :TELEFONE1, :TELEFONE2) RETURNING ID INTO :ID_RET"
                dadosPessoa["ID_RET"] = idPessoaRet

                #cursor.execute trata os dados, protegendo contra injeções
                cursor.execute(sqlInsertPessoa, dadosPessoa)
                #Pega o ID retornado
                idPessoaBytes = idPessoaRet.getvalue()[0]

            sqlInsertPaciente = \
                "INSERT INTO PACIENTE (PESSOA, SEXO, NASCIMENTO, OBITO, COR, PESO, TELEFONE_EMERGENCIA1, TELEFONE_EMERGENCIA2) " \
                "VALUES (:ID_PESSOA_BYTES, :SEXO, :NASCIMENTO, :OBITO, :COR, :PESO, :TELEFONE_EMERGENCIA1, :TELEFONE_EMERGENCIA2)"

            dadosPaciente["ID_PESSOA_BYTES"] = idPessoaBytes

            #cursor.execute trata os dados, protegendo contra injeções
            cursor.execute(sqlInsertPaciente, dadosPaciente)
            #Chama commit na base de dados, salvando os dados por definitivo
            conn.commit()

            print(f"\n\nPaciente ID = {BinParaHex(idPessoaBytes)} registrado com sucesso!")
            #Instrui o usuário à orientar o Paciente
            print("Caso o paciente tenha interesse em se tornar doador de órgãos, informe-o sobre os próximos passos:")
            print("- Ele pode manifestar sua vontade conversando com a família, que é a responsável pela autorização final.")
            print("- É recomendado esclarecer dúvidas com a equipe médica ou com o serviço de orientação do hospital.")
            print("- Se desejar, o paciente pode solicitar materiais explicativos sobre doação de órgãos.")
            print("- Reforce que a decisão é voluntária e pode ser alterada a qualquer momento.\n\n")

#with conn -> realiza rollback automático quando sai do seu bloco
except oracledb.Error as e:
    print(f"\nErro oracle: {e}\n")
except Exception as e:
    print(f"\nErro: {e}\n")
```

Figura 4: *Inserções nas tabelas PESSOA e PACIENTE*

Esse comando materializa de fato a criação de uma nova pessoa no sistema. O uso de *RETURNING* na variável *sqlInsertPessoa* é fundamental, pois permite passar imediatamente o ID gerado para o próximo passo, que depende da integridade referencial entre PESSOA e PACIENTE. Assim, evita-se a necessidade de consultas adicionais apenas para recuperar esse valor.

Para a conexão com o SGDB, utiliza-se um *connection pool* fornecido pela biblioteca oracledb, criado antes mesmo das execuções dos comandos SQL. Ela garante que sempre haverá no mínimo 1 e no máximo 2 conexões ativas, permitindo operações rápidas em conexões pré-estabelecidas.

```python
if __name__ == "__main__":
    #Abre o .env
    load_dotenv()

    #Pega os dados do .env
    db_user = os.getenv("user")
    db_pass = os.getenv("password")
    db_host = os.getenv("host")
    db_port = os.getenv("port")
    db_service = os.getenv("service_name")

    #Se algum dado estiver faltando (ou o .env em si)
    if not all([db_user, db_pass, db_host, db_port, db_service]):
        print("\n[ERRO] Arquivo .env incompleto!\n")
        print("Verifique: user, password, host, port, service_name\n")
        exit()

    dsn = oracledb.makedsn(host=db_host, port=db_port, service_name=db_service)

    pool = None
    try:
        print("Conectando ao banco de dados...")
        pool = oracledb.create_pool(
            user=db_user,
            password=db_pass,
            dsn=dsn,
            min=1,
            max=2,
            increment=1
        )
        print("Sistema iniciado com sucesso!\n")
```

Figura 5: *Código de criação do pool*

A chamada *pool.acquire()* é utilizada sempre que uma função precisa interagir com o banco, garantindo que toda operação receba rapidamente uma conexão própria e isolada. Esse mecanismo traz dois benefícios importantes: primeiro, cada conjunto de comandos SQL é executado em uma “sessão” nova, reduzindo riscos de *timeout* e agregando comandos relacionados; segundo, o gerenciamento transacional torna-se mais simples, pois o *pool* automaticamente faz um *rollback* automático ao sair da “sessão” (ao sair do bloco da conexão). Este último benefício dispensa a necessidade de chamar o *rollback* explicitamente em casos de erros, restando apenas realizar o *commit* quando a operação é bem-sucedida. Assim, se uma exceção ocorre antes do *commit*, a própria liberação da “sessão” impede que alterações parciais sejam persistidas.

Por fim, a função *SelectPessoa* implementa uma consulta SQL parametrizada que serve como interface de busca no protótipo da aplicação. Trata-se de uma consulta extremamente configurável, pois aceita múltiplos parâmetros opcionais e monta uma filtragem completa de acordo com os campos informados pelo usuário.

```
#Todos os valores que estiverem como None serão ignorados pelo trecho ":atributo IS NULL"
sqlSelectPessoa = "SELECT * FROM PESSOA WHERE " \
        "(:idPessoa IS NULL OR ID = :idPessoa) " \
    "AND (:cpf IS NULL OR CPF = :cpf) " \
    "AND (:nome IS NULL OR NOME LIKE '%' || :nome || '%') " \
    "AND (:estado IS NULL OR ESTADO = :estado) " \
    "AND (:cidade IS NULL OR CIDADE = :cidade) " \
    "AND (:bairro IS NULL OR BAIRRO = :bairro) " \
    "AND (:rua IS NULL OR RUA = :rua) " \
    "AND (:numero IS NULL OR NUMERO = :numero) " \
    "AND (:telefone1 IS NULL OR TELEFONE1 = :telefone1) " \
    "AND (:telefone2 IS NULL OR TELEFONE2 = :telefone2)"

dados = {
    "idPessoa": idPessoa,
    "cpf": cpf,
    "nome": nome,
    "estado": estado,
    "cidade": cidade,
    "bairro": bairro,
    "rua": rua,
    "numero": numero,
    "telefone1": telefone1,
    "telefone2": telefone2
}
```

Figura 6: *Código de definição do comando de Select*

```
try:
    #Pega uma conexão com o BD
    with pool.acquire() as conn:
        #Cria um cursor pra conexão
        with conn.cursor() as cursor:
            #cursor.execute trata os dados, protegendo contra injeções
            cursor.execute(sqlSelectPessoa, dados)

            rows = cursor.fetchall()
            #Pega o nome das colunas
            cols = [desc[0] for desc in cursor.description]

            #Converte a primeira coluna de toda linha (o ID) de binário para hexadecimal
            #[...] + row[1:] -> concatena o resultado da função com o restante da linha
            hexRows = [[BinParaHex(row[0])] + list(row[1:]) for row in rows]

            #Imprime a tabela obtida
            print(f"\n==== Tabela Pessoa ====")
            print(tabulate(hexRows, headers=cols, tablefmt="psql"))

            #Print de separação, para facilitar a legibilidade
            print("")

except oracledb.Error as e:
    print(f"\nErro oracle: {e}\n")
except Exception as e:
    print(f"\nErro: {e}\n")
```

Figura 7: *Código de execução do comando de Select*

Assim como nos demais comandos SQL da aplicação, todo o processamento das entradas é realizado pela função *cursor.execute()* do oracledb. Esta função indexa os dados passados para o comando desejado, transformando os valores para os tipos de dados ideais para a execução. Isto garante que qualquer comando em SQL malicioso enviado pelo usuário seja tratado como uma string, o que impede a realização de ataques de *SQL Injection*.

4.3. Consultas

Consulta 1 – Doadores compatíveis para receptores de prioridade máxima

```sql
--Procura todos os doadores disponíveis para os receptores com prioridade máxima,
--avaliando o tipo sanguíneo encontrado nos exames de cada um para verificar a compatibilidade do receptor com o doador.
--Se um deles não tiver realizado o exame, eles não serão considerados pela bsuca
SELECT DISTINCT R.TIPO_ORGAO, R.RECEPTOR, D.DOADOR
FROM RECEPTOR_ESPERA R
JOIN EXAME E1 ON R.RECEPTOR = E1.PACIENTE
JOIN DOADOR_DOA D ON R.TIPO_ORGAO = D.TIPO_ORGAO
JOIN EXAME E2 ON D.DOADOR = E2.PACIENTE
WHERE
--Verifica se os tipos sanguíneos são compatíveis
(
        --Receptor == O
        --Doador == O
        (E1.RESULTADO LIKE '% O_,%'
         AND E2.RESULTADO LIKE '% O_,%')

        --Receptor == A
        --Doador == O ou A
    OR  (E1.RESULTADO LIKE '% A_,%'
         AND (
                E2.RESULTADO LIKE '% O_,%'
             OR E2.RESULTADO LIKE '% A_,%'
             )
        )

    --Receptor == B
    --Doador == O ou B
    OR  (E1.RESULTADO LIKE '% B_,%'
         AND (
                E2.RESULTADO LIKE '% O_,%'
             OR E2.RESULTADO LIKE '% B_,%'
             )
        )

    --Receptor == O
    --Doador == O ou A ou B ou AB
    OR  (E1.RESULTADO LIKE '% AB_,%'
         AND (
                E2.RESULTADO LIKE '% O_,%'
             OR E2.RESULTADO LIKE '% A_,%'
             OR E2.RESULTADO LIKE '% B_,%'
             OR E2.RESULTADO LIKE '% AB_,%'
             )
        )
)
AND
    --Compara o Rh
    NOT (E1.RESULTADO LIKE '%-,%' AND E2.RESULTADO LIKE '%+,%')
    --Verifica se o receptor tem prioridade máxima
AND R.PRIORIDADE = 1;
```

Figura 8: *Select de doadores compativeís*

Essa consulta tem como objetivo identificar, para cada receptor em situação de prioridade máxima, quais doadores do mesmo órgão são compatíveis do ponto de vista sanguíneo. Para isso, as tabelas RECEPTOR_ESPERA e DOADOR_DOA são unidas a dois exames laboratoriais distintos: E1, associado ao receptor, e E2, associado ao doador, ambos obtidos a partir da tabela EXAME. A tabela DOADOR_DOA

também faz a ponte entre o tipo de órgão desejado pelo receptor e os doadores disponíveis para aquele mesmo tipo.

A lógica de compatibilidade do tipo sanguíneo segue as regras padrão do sistema ABO: receptores do tipo O só podem receber de doadores O; receptores A podem receber de O ou A; receptores B podem receber de O ou B; e receptores AB podem receber de qualquer tipo (ainda assim, checa-se o exame E2 para que seja um que contenha o tipo sanguíneo e não outro exame não relacionado, evitando repetição de tuplas). Essas regras são implementadas por meio de uma combinação de condições LIKE sobre o resultado do exame, que procura o tipo sanguíneo dentre os resultados. Além disso, a consulta verifica a compatibilidade do fator Rh, proibindo explicitamente o caso em que o receptor é Rh negativo e o doador é Rh positivo. Por fim, um filtro adicional restringe o conjunto de receptores àqueles cuja prioridade é igual a 1, ou seja, os casos mais críticos da fila.

O resultado traz, para cada combinação, tipo de órgão em questão, o identificador do receptor e o identificador do doador, permitindo que a equipe da CIHDOTT tenha uma visão clara dos pares doador–receptor clinicamente viáveis em situações de maior urgência.

Consulta 2 – Número de óbitos durante cirurgias por hospital

```sql
--Lista o número de óbitos durante cirurgias de cada hospital
SELECT H.NOME,
--Conta o número de pessoas cujo horário de óbito
--está dentro da janela de alguma cirurgia em que elas eram pacientes
COUNT(
    CASE
        WHEN P.OBITO BETWEEN C.DATA_HORARIO_INICIO AND C.DATA_HORARIO_TERMINO
        THEN 1
    END
) AS OBITOS
FROM HOSPITAL H
--LEFT JOIN para preservar, no select, hospitais que não realizaram cirurgia alguma
LEFT JOIN CIRURGIA C ON H.ID = C.HOSPITAL
LEFT JOIN PACIENTE P ON P.PESSOA = C.PACIENTE
GROUP BY H.NOME
ORDER BY OBITOS DESC;
```

Figura 9: *Select de número de óbitos em cirurgia por hospital*

Nesta consulta busca-se quantificar, para cada hospital, o número de óbitos que ocorreram durante o intervalo de tempo de alguma cirurgia em que o paciente estava sendo operado. A partir da tabela HOSPITAL, realiza-se uma junção com a tabela CIRURGIA para associar cada procedimento ao hospital onde foi realizado e, em seguida, uma junção com a tabela PACIENTE, de forma a relacionar o registro de óbito ao paciente que participou daquela cirurgia.

Utiliza-se *LEFT JOINs* pois é necessário que os hospitais que não tem nenhuma cirurgia registrada no sistema também apareçam na saída (no caso, apareceram como tendo 0 óbitos). A contagem é feita com uma expressão COUNT aplicada sobre um CASE que retorna o valor 1 apenas quando o horário de óbito do paciente está entre o horário de início e o de término da cirurgia correspondente; fora desse intervalo, o CASE retorna NULL, que não é contabilizado pelo COUNT. Assim, o valor agregado representa exatamente o número de pacientes que faleceram durante a janela de alguma cirurgia em cada hospital. A consulta agrupa os resultados pelo nome da instituição e ordena pela contagem de óbitos em ordem decrescente.

Do ponto de vista gerencial, essa funcionalidade permite identificar hospitais com maior incidência de óbitos intraoperatórios, o que pode indicar a necessidade de investigação de protocolos, avaliação de risco cirúrgico ou alocação de recursos adicionais.

Consulta 3 – Doadores que doaram todos os tipos de órgãos possíveis

```sql
-- Lista os doadores que doaram todos os órgãos possíveis
SELECT DISTINCT C.PACIENTE
FROM CIRURGIA C
WHERE NOT EXISTS
(
    (
        SELECT T.NOME
        FROM TIPO_ORGAO T
    )
    MINUS
    (
        SELECT O.TIPO
        FROM CIRURGIA C2 JOIN ORGAO O
        ON C2.ID = O.COLETA
        WHERE C2.PACIENTE = C.PACIENTE
    )
);
```

Figura 10: *Select de pacientes que, ao longo da vida, doaram todos os tipos de órgãos possíveis*

Esta consulta implementa a ideia de divisão relacional: o objetivo é encontrar quais pacientes já doaram todos os tipos de órgãos cadastrados na tabela TIPO_ORGAO. Para isso, parte-se da tabela CIRURGIA, considerando os pacientes que participaram nas cirurgias de coleta, e utiliza-se uma cláusula NOT EXISTS combinada com o operador MINUS.

O primeiro conjunto dentro do NOT EXISTS é simplesmente o conjunto de todos os tipos de órgãos existentes no sistema, obtido por um SELECT sobre TIPO_ORGAO. O segundo conjunto corresponde aos tipos de órgãos efetivamente coletados daquele paciente, calculado a partir da junção entre CIRURGIA e ORGAO, restringindo-se às cirurgias de coleta para o mesmo paciente considerado na linha externa. A expressão “A MINUS B” devolve os tipos de órgãos que existem na base, mas que não aparecem em nenhuma coleta daquele paciente.

Ao aplicar o NOT EXISTS, apenas os conjuntos vazios serão aceitos, que é exatamente o critério de divisão: o doador em questão doou pelo menos um órgão de cada tipo cadastrado.

Consulta 4 – Pacientes que doaram órgãos que eles próprios haviam recebido

```sql
--Checa se existem pessoas que doaram órgãos que elas receberam em uma cirurgia anterior. Os rins são desconsiderados nessa busca
SELECT C.PACIENTE, O.TIPO, O.LADO
FROM CIRURGIA C
JOIN ORGAO O ON C.ID = O.COLETA
JOIN ORGAO O2 ON O2.TIPO = O.TIPO AND O2.LADO = O.LADO
JOIN CIRURGIA C2 ON C2.ID = O2.RECEPCAO AND C2.PACIENTE = C.PACIENTE
WHERE C.DATA_HORARIO_INICIO > C2.DATA_HORARIO_TERMINO
AND O.TIPO <> 'RIM';
```

Figura 11: *Select de pacientes que doaram órgãos que haviam recebido de outro transplante*

O propósito desta consulta é detectar situações em que um paciente doa um órgão do mesmo tipo e lado que ele próprio já havia recebido em uma cirurgia anterior, ignorando doações/recepções de rins, uma vez que eles são simétricos e se tornaria inviável garantir que foi recebido do mesmo lado.

A lógica parte da tabela CIRURGIA associada à tabela ORGAO, considerando as cirurgias em que o órgão foi coletado. Em seguida, a *query* introduz um segundo grupos de órgãos, que representa o órgãos de mesmo lado e mesmo tipo. A partir daí, uma nova junção com CIRURGIA garante que estes órgãos foram usados para uma recepção anterior, a qual envolveu o mesmo paciente, de modo que se constrói a cadeia "paciente recebe órgão – mais tarde doa um órgão igual". A condição temporal C.DATA_HORARIO_INICIO $>$ C2.DATA_HORARIO_TERMINO impõe que a cirurgia de coleta aconteça somente depois do término da cirurgia de recepção, respeitando a ordem cronológica dos eventos.

Por fim, é aplicado um filtro que exclui órgãos do tipo "RIM", uma vez que, nesse caso específico, o lado do órgão não é confiável para fins de comparação: rins podem ser reposicionados cirurgicamente, de modo que um rim direito pode ser implantado à esquerda e vice-versa, o que inviabiliza a verificação baseada no atributo LADO e tornaria a consulta incorreta para esse tipo de órgão. O resultado da consulta retorna o identificador do paciente e o tipo de órgão.

Consulta 5 – Cirurgias realizadas sem autorização SNT válida para o órgão

```sql
--Seleciona hospitais que realizaram cirurgias para os quais não estavam autorizados
SELECT H.NOME, O.TIPO, TO_CHAR(C.DATA_HORARIO_INICIO, 'YYYY/MM/DD HH24:MI:SS') AS DATA_HORARIO_INICIO, C.TIPO
FROM HOSPITAL H
JOIN CIRURGIA C ON C.HOSPITAL = H.ID
JOIN ORGAO O ON C.ID = O.COLETA OR C.ID = O.RECEPCAO
LEFT JOIN AUTORIZACAO_HOSPITAL A ON H.ID = A.HOSPITAL
AND INSTR(A.AUTORIZACAO_SNT, O.TIPO) > 0
AND C.DATA_HORARIO_INICIO < A.VALIDADE_AUTORIZACAO
WHERE A.HOSPITAL IS NULL;

--Nota: pode acontecer que um hospital realize uma cirurgia não autorizada, mas renove sua autorização depois
--Nesse caso, a informação de que a cirurgia foi realizada indevidamente será perdida da base de dados
```

Figura 12: *Select de realizadas sem autorização SNT válida para o órgão*

Esta consulta busca identificar cirurgias realizadas por hospitais sem a autorização adequada do Sistema Nacional de Transplantes para o órgão envolvido na operação na data em que o procedimento ocorreu. A partir da tabela HOSPITAL, a consulta estabelece junções com CIRURGIA e ORGAO, permitindo recuperar, para cada procedimento, o órgão manipulado. Em seguida, é aplicada uma junção externa com AUTORIZACAO_HOSPITAL, de forma a comparar o tipo de órgão tratado na cirurgia com o conteúdo das autorizações registradas para aquele hospital, além de verificar se a validade da autorização abrangia o período em que a cirurgia foi realizada. Quando não há correspondência, o registro retorna com valores nulos para o hospital na tabela de autorização. A cláusula final filtra exatamente essas ocorrências, preservando apenas as cirurgias que foram executadas sem permissão para aquele tipo de procedimento. Ou seja, é implementado, nessa consulta, um *LEFT ANTI JOIN* para achar os hospitais que realizaram cirurgias sem autorização.

O resultado exibe o nome do hospital, o órgão tratado, a data e horário de realização e o tipo da cirurgia, fornecendo uma visão direta de possíveis violações. Infelizmente, há um aspecto da modelagem dessa base de dados que limita a atuação dessa consulta: se o hospital renovar sua autorização após realizar as cirurgias indevídas, a data de validade da autorização será incrementada, e a informação de que a cirurgia foi realizada indevidamente se perde da base de dados. A correção deste problema requeriria uma reformulação da tabela AUTORIZACAO_HOSPITAL, incluindo sua chave primária.

5. Conclusão

O desenvolvimento deste projeto permitiu ao grupo vivenciar de forma prática e integrada os principais conceitos da disciplina de Bases de Dados. Criar um sistema completo tornou o aprendizado mais claro

e mostrou como cada etapa da disciplina se conecta para formar uma solução real.

A elaboração do Modelo Entidade–Relacionamento foi o primeiro grande passo. Apesar de parecer simples no início, logo ficou evidente que as decisões de modelagem têm impacto direto nas fases seguintes. Foram consideradas diferentes alternativas, feitas revisões e realizados vários ajustes até chegar a uma estrutura que representasse bem as regras do domínio. Esse processo foi fundamental para fortalecer a habilidade de transformar problemas do mundo real em modelos formais consistentes.

Com essa base estabelecida, a criação do modelo relacional ocorreu de maneira mais tranquila, embora exigisse cuidado para manter integridade, clareza e evitar redundâncias. Já o desenvolvimento das consultas SQL foi uma das etapas mais enriquecedoras, pois demandou soluções adequadas ao contexto do projeto e alinhadas aos requisitos da disciplina, incluindo consultas mais complexas e a aplicação do conceito de divisão relacional.

A implementação do protótipo trouxe desafios importantes, especialmente na integração entre a aplicação e o banco de dados e no tratamento dos dados inseridos pelos usuários. Essa etapa mostrou como a base teórica da disciplina permitiu ao grupo enfrentar problemas práticos, lidar com erros e garantir que os dados fossem processados corretamente.

No geral, o projeto proporcionou uma experiência completa e alinhada aos objetivos da disciplina. A união entre teoria e prática permitiu entender não apenas os conceitos fundamentais de bancos de dados, mas também sua relevância em aplicações reais. Observou-se um avanço significativo tanto nas competências técnicas quanto na capacidade de analisar, modelar e implementar soluções. A disciplina teve papel essencial nesse processo, preparando o grupo para trabalhar com sistemas de informação mais complexos em contextos acadêmicos e profissionais, e reforçando a importância de uma boa modelagem para o desenvolvimento de aplicações robustas.