Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

SSC0240 - Bases de Dados
Prof. Dra. Elaine Parros M. de Sousa

Projeto Conectec

Integrando comunidades ao meio digital

Fernando Gonçalves Campos – 12542352
Bernardo Rodrigues Tameirão Santos – 12733212
Fernando César Lopes Barbosa Filho – 10260559
José Pedro Cioni do Carmo – 12623988
Samuel Figueiredo Veronez –12542626

Sumário

1 Primeira entrega: Modelo Conceitual

1.1 Objetivo do projeto

Apesar da crescente importância das tecnologias digitais no cotidiano da sociedade, existe uma grande disparidade entre o avanço tecnológico e o acesso das comunidades carentes a essas tecnologias. O objetivo da plataforma proposta é integrar tais comunidades a essa nova realidade digital – permitindo o acesso tanto a aparelhos para uso como a professores capacitados em não apenas ensinar a utilização do aparelho em si como também as diversas funcionalidades nele presentes.

1.2 Levantamento de requisitos

Para as entidades da modelagem temos os seguintes requisitos:

- *Centro*: Cada centro possui nome, endereço e número de docentes. As salas disponíveis em um centro são divididas em blocos.

- ***Bloco:*** Cada bloco possui um número próprio que, junto com o nome do centro ao qual ele pertence, serve como identificador do bloco.

- ***Sala***: Uma sala pode ser especializada em sala de computadores ou sala de eventos. As salas de computadores possuem equipamentos, e também podem se especializar em salas de aula e salas de uso livre. As salas de evento podem ser reservadas para apresentações, conferências e eventos semelhantes. Cada sala possui bloco, número e ocupação máxima.

- ***Pessoa***: cada pessoa possui CPF(chave), nome, nascimento, sexo, telefone e cargo. Coordenador, professor, aluno e técnico são possíveis especializações de pessoa.

- ***Disciplina***: Uma disciplina é ministrada por um professor em uma sala predeterminada, deve ser possível ocorrer com os equipamentos disponíveis em sua sala designada e vários alunos podem participar de uma aula. Cada disciplina possui horário de início e de fim, duração e número máximo de alunos.

- ***Evento***: Um evento é criado e precisa ter ao menos uma sala reservada, Ele é identificado de acordo com as salas e os horários em que foi reservado. Todo evento possui uma proposta (nome e descrição detalhada), que é o método utilizado para divulgar o evento para outras pessoas. Pessoas podem se inscrever para eventos que estão marcados como sendo públicos.

- ***Equipamento***: Um equipamento possui nome, número de série e condição, podendo ter problemas que exijam manutenção por parte de um técnico.

- ***Coordenador***: Um coordenador é uma pessoa responsável pela organização do projeto e deve garantir que os professores sejam capacitados, assim, ele pode remover/adicionar professores.

- ***Professor***: Um professor pode criar novas disciplinas, editar suas aulas já existentes, e alterar a condição dos equipamentos.

- ***Aluno***: Um aluno pode inserir/remover participação em uma aula. Como também pode atuar como monitor de uma monitoria, um mesmo aluno não pode ser monitor de mais de uma disciplina.

- ***Monitoria***: A monitoria é composta pela disciplina à qual ela pertence e pelo aluno monitor, as monitorias são diferenciadas pelo aluno monitor. Uma

monitoria possui pelo menos uma sala de aula, juntamente com os horários nos quais essa monitoria acontecerá nesta sala.

- ***Técnico***: Um técnico é responsável pela manutenção dos aparelhos das salas, podendo alterar a condição dele no sistema para refletir o estado de funcionamento.

1.3 Principais operações

Para as todas as Pessoas, temos as seguintes operações:

- Inscrever-se em eventos.
- Reservar salas para eventos.

Para as diferentes especializações de *Pessoa*, temos as seguintes operações:

Aluno:

- Matricular-se em uma aula.
- Buscar aulas, podendo filtrar por comunidade, tema
- Oferecer-se como monitor de uma disciplina

Professor:

- Inserir, remover, editar aulas que ministra
- Buscar salas, podendo filtrar por comunidade, equipamentos.
- Buscar equipamentos, podendo filtrar por sala.
- Alterar a condição de equipamentos.

Coordenador:

- Inserir/remover professores
- Buscar professores, podendo filtrar por aula
- Supervisionar professores
- Coordenar o centro

Técnico:

- Alterar a condição de equipamentos através da manutenção.
- Buscar equipamentos, podendo filtrar por condição e sala.

1.4 Restrições de integridade

Temos as seguintes restrições:

- É responsabilidade da aplicação garantir que não haja colisão de horário entre aulas/monitorias em uma mesma sala durante a criação de uma nova aula.
- A aplicação deve impedir a participação de um aluno em uma aula lotada.
- Uma pessoa não pode reservar mais de uma sala em horários que se sobreponham.
- Ciclo ***Aluno → Disciplina***:

  Há um ciclo entre as entidades *Aluno* e *Disciplina*, tomando os relacionamentos *Matricula* e *Monitora*. Esse ciclo não pode ser quebrado, uma vez que ambos representam ações de dois alunos distintos em disciplina e, portanto, necessita da utilização de dois relacionamentos diferentes entre ambas entidades. Os alunos podem se relacionar com a disciplina sendo um monitor ou se matriculando em uma aula. Na nossa implementação, no entanto, não é permitido que um aluno dê uma monitoria sem antes ter concluído a disciplina. Dessa forma, o ciclo poderá gerar uma inconsistência nos dados que deverá ser resolvida na implementação.

- Ciclo ***Pessoa → Evento***:

  Existe um ciclo entre as entidades *Pessoa* e *Evento*, ao serem considerados os relacionamentos Reserva e Inscreve. Esse ciclo não pode ser rompido, visto que ambas as partes simbolizam ações distintas de pessoas em *Evento*. Não foi encontrada nenhuma inconsistência, já que o criador do evento poderá se inscrever no próprio evento se quiser.

- Ciclo ***Sala de aula → Disciplina***:

  Temos um ciclo entre as entidades *Sala de Aula* e *Disciplina*, ao serem considerados os relacionamentos *Leciona* e *Monitora*. Esse ciclo não pode ser quebrado, uma vez que ambos representam salas que podem ser diferentes: uma utilizada para monitoria e outra para lecionar a disciplina. Para não ter inconsistências, é necessário que não haja conflito de horário, como mencionado anteriormente.

- Ciclo ***Sala de aula → Aluno***:

  Há a presença de um ciclo entre as entidades *Sala de Aula* e *Aluno*, podendo haver inconsistência envolvendo o relacionamento *Monitoria* com a entidade

Disciplina e o relacionamento *Matrícula*. Na implementação devemos impedir que um mesmo aluno esteja envolvido em salas diferentes num mesmo horário, seja assistindo aula ou dando monitoria – como já mencionado.

- Ciclo ***Sala de aula → Coordenado → Professor***:

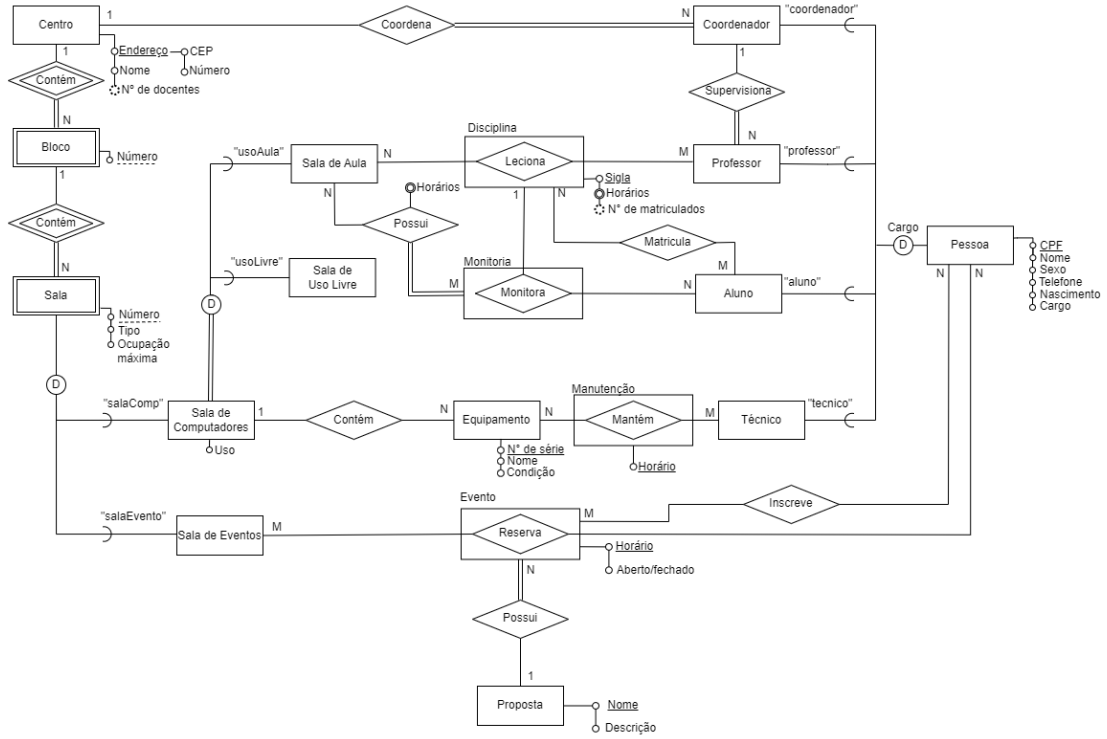
  Há a presença de um ciclo entre as entidades *Sala de Aula* e *Coordenador* e *Professor* podendo haver inconsistência envolvendo a entidade *Sala de Aula*, na qual um professor iria lecionar uma disciplina em uma sala cujo centro ao qual pertence seja diferente do centro coordenado pelo supervisor do professor. Na implementação devemos impedir que um professor lecione disciplinas em salas “inválidas”.

- Ciclo ***Evento → Coordenador/Professor***:

  Há a presença de um ciclo entre as entidades *Evento* e *Coordenador/Professor*, no entanto este ciclo não causa inconsistência, tendo em vista que um coordenador/professor pode criar/participar de um evento em um centro no qual eles não trabalham.

2 Diagrama Entidade-Relacionamento (DER)

Obtemos, a partir das descrições fornecidas no capítulo anterior, o seguinte diagrama:



Assume-se que cada centro possui um nome único. Dessa forma o nome pode ser usado como chave

Há a presença do ciclo Aluno -> Disciplina que pode gerar inconsistência de dados. Um aluno não pode dar monitoria de uma disciplina que ele não tenha feito.

Deve-se impedir um aluno de se matricular numa sala lotada, ou uma pessoa em um evento lotado

Deve-se garantir que não haja conflito de horário ou sala na criação de uma disciplina, monitoria ou evento.

Um aluno pode ser monitor de apenas uma disciplina, ele pode realizar monitoria em quantas salas quiser. Uma disciplina pode ter mais de um monitor

A chave de um evento é a sala na qual ele acontece e o horário em que acontece.

A chave de uma disciplina é a sua sigla.

A chave de manutenção é a data em que acontece e o nº de série do equipamento

A chave de monitoria é o aluno

Um horário é composto por data, horário de início e horário de encerramento

A proposta seria o meio de divulgar o evento para outras pessoas

Salas reservadas para um evento podem se distinguir entre abertas e fechadas:

-Aberto significa que pessoas podem se inscrever para o evento

-Fechado pessoas não podem se inscrever nesses horários.

Isso possibilita que a organização do evento possa reservar a sala em horários que não serão divulgados, alguns eventos precisam de tempo antes do início para poder se organizar

Uma mesma proposta pode ocorrer em centros diferentes

Garantir que as reservas sejam válidas (possuam um propósito válido) é algo que deve ser resolvido na implementação.

3 Mudanças relacionadas à primeira entrega

3.1 Documentação

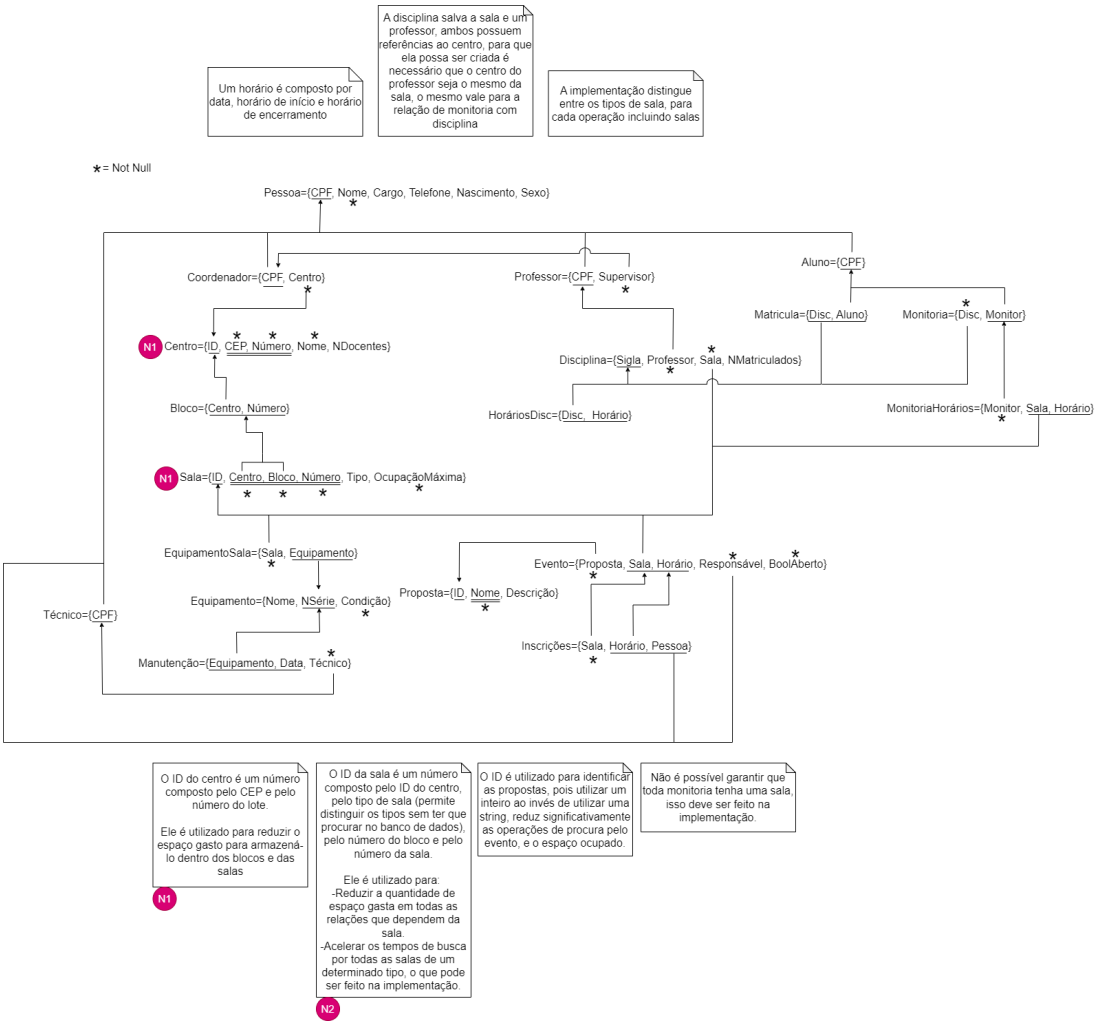
- Foram documentados todos os relacionamentos que não haviam sido descritos.

3.2 Diagrama MER

- Foi corrigida a notação que estava sendo utilizada nos relacionamentos.
- O relacionamento monitoria foi convertido em um agregado de um relacionamento entre aluno e disciplina que se relaciona com salas de aula.
- A manutenção de equipamentos foi convertida em um agregado que possui como chave a data em que ocorreu e o equipamento que recebeu a manutenção.
- O evento foi modificado para permitir que uma mesma proposta pudesse ser realizada em mais de uma sala e em vários horários diferentes. Além de permitir que seja feita a reserva em horários diferentes dos que o evento irá ocorrer, possibilitando que eventos que precisam de mais tempo para preparativos possam ter a sala disponível para esse tempo.
- Professor passou a ter participação total no relacionamento *Supervisiona*.
- As informações salvas para o *Centro* foram levemente modificadas, e a sua chave passou a ser *Endereço* em vez de *Nome*.
- Sala passou a ser uma entidade fraca do *Bloco*.

4 Segunda Entrega: Modelo Relacional

4.1 Esquema Relacional



4.2 Justificativa de escolhas do modelo relacional

4.2.1. Especialização da entidade Pessoa

Solução adotada: Foi criada uma tabela para cada especialização de *Pessoa*, e uma tabela que salva todas as pessoas.

Vantagens: Possibilita que pessoas especializadas tenham atributos próprios, sem fazer com que, para os outros tipos de pessoas esses atributos sejam

nulos. Além disso, permite que, posteriormente, sejam adicionados atributos próprios, caso haja necessidade, sem ter que copiar os dados de todas as pessoas, apenas das pessoas do tipo escolhido.

Desvantagens: Em situações nas quais se é relevante obter tanto os atributos gerais, como os atributos próprios, existe um custo maior na busca, dado que é necessário buscar em duas tabelas em vez de uma. Além disso, possuir múltiplas tabelas aumenta o gasto de memória, por ter que, efetivamente, consumir o dobro de memória para salvar o *CPF*. Importante também dizer que esse método não garante a disjunção.

Solução Alternativa: Uma alternativa seria ignorar as especializações de pessoas e concentrar todas em uma única tabela. Essa alternativa foi desconsiderada, principalmente, para evitar desperdício de espaço, por conta de reservar memória para guardar valores como *Centro do Coordenador* e *Supervisor do Professor* que seriam nulos para a maioria das pessoas.

4.2.2. Relacionamento “Contém” entre Centro e Bloco - 1:N

Solução adotada: A tabela relativa a *Bloco* contém uma chave estrangeira de *Centro*, que juntamente ao número do bloco compõem a chave primária do *Bloco*.

Vantagens: Garante a participação total de *Bloco* no relacionamento.

Desvantagens: Não foram encontradas desvantagens nessa solução.

Solução Alternativa: *Bloco* poderia ser um campo não nulo e não compor a chave primária, entretanto essa solução dependeria do atributo *Número* único entre todos os blocos, o que julgamos como menos intuitivo.

4.2.3. Relacionamento “Coordena” entre Centro e Coordenador - 1:N

Solução adotada: A tabela relativa a *Coordenador* possui um atributo não nulo que a liga à chave primária de *Centro*.

Vantagens: Garante a participação total de *Coordenador* no relacionamento.

Desvantagens: Não foram encontradas desvantagens nessa solução.

Solução Alternativa: Poderia ter sido criada uma tabela adicional que relacione *Coordenador* e *Centro*. Essa solução não garantiria a participação total e dependeria da criação de uma tabela adicional.

4.2.4. Relacionamento “Contém” entre Bloco e Sala - 1:N

Solução adotada: A tabela *Sala* possui chaves externas para os elementos que compõem a chave primária de *Bloco*, que juntamente ao *Número* da sala compõe a chave secundária, todos estes valores são não nulos. A chave primária de Sala é um ID sintético.

Vantagens: O uso do ID sintético promove armazenamento mais eficiente nos relacionamentos que dependem de *Sala*. Enquanto os atributos *Centro* e *Bloco*, juntamente ao *Número* da *Sala*, não nulos, garantindo a participação total da *Sala* no relacionamento.

Desvantagens: O uso de um ID sintético causa uma perda semântica da chave da Sala. Isso é ruim, pois cria a necessidade de uma junção adicional quando são necessárias informações referentes ao *Centro*, *Bloco* ou *Número* de uma *Sala* referenciada em outras tabelas.

Solução Alternativa: Utilização dos atributos que compõem a chave primária de *Bloco* e o número como chave-primária. Entretanto foi desconsiderada pois sala participa de diversos relacionamentos, o que implicaria na replicação desses campos ao longo de várias tabelas, o que seria ineficiente em relação ao armazenamento.

4.2.5. Especialização da entidade Sala

Solução adotada: Foi mapeada apenas uma tabela *Sala*, sendo suas especializações tratadas por meio da existência do atributo *Tipo*, esse método foi considerado plausível já que as especializações não contém atributos extras.

Vantagens: Não é necessário a criação de múltiplas tabelas para distinguir entre as salas.

Desvantagens: Não é garantida a especialização total para as salas de computadores. Realizar a busca por salas de um tipo específico é mais custoso, tendo em vista que as estão sendo armazenadas juntas. Este problema é amenizado pelo método de formação do ID [utiliza o tipo para tornar a ordenação mais conveniente *(Centro -> Tipo -> Bloco -> Sala*)].

Também vai ser necessário um tratamento adicional pela aplicação para garantir que cada sala tenha apenas os usos para os quais foi designada.

Solução Alternativa: Seria possível criar várias tabelas, uma para cada tipo de sala e uma tabela para armazenar todas as salas. Foi desconsiderada pelo aumento no número de tabelas, e por ser redundante já que os dados armazenados para todos os tipos de sala são os mesmos.

4.2.6. Relacionamento “Contém” entre Sala de Computadores e Equipamento - 1:N

Solução adotada: Criação de uma tabela adicional que associa *Sala* e *Equipamento*.

Vantagens: Em relação a solução alternativa, não existem campos nulos para Equipamentos que não estão associados à *Salas de Computadores*.

Desvantagens: Foi necessária a adição de uma tabela adicional, o que pode impactar negativamente na performance da busca dos dados dos computadores que pertencem a uma *Sala*.

Solução Alternativa: Armazenar a chave estrangeira de *Sala* de *Computadores* no *Equipamento*. Entretanto, no caso de muitos equipamentos sem sala, gera muitos campos *Sala* com valores nulos.

4.2.7. Relacionamento “Reserva” entre Sala de Eventos e Pessoa - N:M

Solução adotada: Atributo em *Evento* não-nulo que liga à chave primária de Pessoa.

Vantagens: Garante a cardinalidade da relação.

Desvantagens: Não foram encontradas desvantagens.

Solução Alternativa: Um outro caminho seria criar uma entidade com chave primária composta por *Sala*, *Responsável* e *Horário*.

4.2.8. Relacionamento “Inscreve” entre Evento e Pessoa - N:M

Solução adotada: Criação da tabela Inscrições, que relaciona um *Evento* (por meio das chaves estrangeiras *Sala* e *Horário*) e uma *Pessoa*. *Horário* e *Pessoa* compõem a chave primária da relação.

Vantagens: Garante a cardinalidade da relação.

Desvantagens: Criação de uma tabela adicional.

Solução Alternativa: Um outro caminho seria a criação de uma tabela com chave primária composta por chave de *Evento* com chave de *Pessoa*.

4.2.9. Relacionamento “Mantém” entre Equipamento e Técnico - N:M

Solução adotada: Criação de uma tabela, que reúne as informações do relacionamento Mantém com as da agregação *Manutenção*, que associa *Equipamento* e *Técnico*. *Equipamento*, juntamente com o atributo *Data* compõe a chave primária. Técnico deve ser não-nulo.

Vantagens: Reduz o gasto de memória com valores redundantes que já estariam sendo salvos na tabela da entidade agregada.

Desvantagens: Não foram encontradas desvantagens relevantes.

Solução Alternativa: Criação de uma tabela para a relação *Mantém* e uma para a agregação manutenção, entretanto isso seria redundante, visto que as informações dessa agregação e relacionamento podem ser reunidas em apenas uma tabela, resultando em um gasto menor de memória.

4.2.10. Relacionamento “Possui” entre Sala de Aula e Monitoria - N:M

Solução adotada: Criação de uma tabela adicional que relaciona um *Monitor* não-nulo a uma *Sala* e *Horário*. Sala e Horário compõem a chave primária.
Vantagens: Garante que nenhuma monitoria possa acontecer em uma mesma sala e mesmo horário.
Desvantagens: Não garante a participação total de monitoria no relacionamento.
Solução Alternativa: Não foram encontradas alternativas estruturalmente distintas que promovam as características desejadas.

4.2.11. Relacionamento “Leciona” entre Sala de Aula e Professor - N:M

Solução adotada: Os valores do relacionamento são salvos juntamente com os valores da agregação disciplina.

Vantagens: Evita redundância em criar uma tabela apenas para salvar se um professor está lecionando em uma determinada sala de aula, dado que esse valor já é salvo na disciplina.

Desvantagens: Não foram encontradas desvantagens para a utilização desta solução.

Solução Alternativa: Uma alternativa seria a criação de uma tabela que salva dois valores, ambos chaves primárias: sala, professor. Esse método não foi utilizado, pois além de não possuir vantagens, ele também gera problemas como redundância e consistência, e seria necessário o tratamento pela aplicação para mitigar esses problemas.

4.2.12. Relacionamento “Monitora” entre Disciplina e Aluno - 1:N

Solução adotada: Criação de uma tabela *Monitoria*, que relaciona um aluno a uma disciplina não nula.

Vantagens: Garante que um aluno possa ser monitor de apenas uma disciplina, economiza memória evitando salvar valores nulos.

Desvantagens: Aumenta o tempo de busca na situação na qual um aluno pretende saber se é o monitor de alguma disciplina.

Solução Alternativa: Outra solução seria adicionar um valor para a disciplina da qual o aluno é monitor, na tabela dos alunos. Essa solução não foi adotada porque potencialmente a grande maioria dos alunos não seriam monitor de nenhuma disciplina, o que implicaria no armazenamento de diversos valores nulos ao longo da tabela.

4.2.13. Relacionamento “Matrícula” entre Disciplina e Aluno - N:M

Solução adotada: Foi criada uma tabela adicional *Matrícula*, que armazena a relação entre *Disciplina* e *Aluno* (chave composta), desta forma uma *Disciplina* pode ter vários alunos e um aluno pode entrar em mais de uma *Disciplina*.

Vantagens: Evita desperdício de memória por excesso de valores nulos.

Desvantagens: A criação de uma nova tabela aumenta a memória utilizada e aumenta o tempo de busca.

Solução Alternativa: Uma alternativa para essa solução seria salvar as disciplinas junto com o restante dos dados dos alunos (ou o contrário). Essa solução não foi adotada, pois, além de limitar o número de disciplinas que um aluno pode cursar, ela também desperdiça muita memória com valores nulos.

4.2.14. Relacionamento "Supervisiona" entre Coordenador e Professor - 1:N

Solução adotada: A tabela *Professor* contém um atributo estrangeiro de *Coordenador*, não nulo. A chave primária é o *CPF* do professor.

Vantagens: Garante a participação total de *Professor* no relacionamento e dispensa a existência de tabelas adicionais.

Desvantagens: Não foram encontradas desvantagens.

Solução Alternativa: Criação de uma tabela que relaciona *Professor* a *Coordenador*, sendo *Professor* a chave primária e *Coordenador* não nulo. Solução descartada pois exige a criação de uma tabela adicional, além de não garantir a participação total do professor no relacionamento.

4.2.15. Relacionamento “Possui” entre Evento e Proposta - 1:N

Solução adotada: Na tabela *Evento* foi adicionado um campo não nulo Proposta, que representa a proposta do *Evento*.

Vantagens: Garante a participação total do *Evento* no relacionamento.

Desvantagens: Não foram identificadas desvantagens em mapear dessa maneira.

Solução Alternativa: Uma solução alternativa seria criar uma tabela adicional, essa alternativa não foi utilizada, pois, além de criar aumentar o custo de memória do sistema, ela também iria aumentar o tempo de procura pela proposta de um evento.

4.2.16. Atributo Multivalorado: Horários de Disciplinas

Solução adotada: Criação de tabela que associa *Disciplina* a um atributo Horário, ambos fazem parte da chave primária desta tabela.

Vantagens: Permite que uma mesma disciplina possua quantos horários forem necessários. Além disso, evita desperdício de memória, salvando muitos valores nulos.

Desvantagens: Criar uma nova tabela pode ser prejudicial para o desempenho, dado que se torna necessário realizar operações de busca sempre que se quiser saber quais os horários de uma disciplina.

Solução Alternativa: Uma solução alternativa seria salvar os horários de uma disciplina na mesma tabela que salva os demais atributos, definindo uma quantidade máxima de horários. Essa solução não foi adotada, pois, como a quantidade de horários de uma disciplina não é fixa, essa abordagem iria permitir que hajam muitos valores nulos sendo salvos, além de possibilitar um cenário em que uma disciplina precisa de mais horários do que o banco de dados permite.

4.2.17. Atributo Multivalorado: Horários de Monitoria

(Já solucionado por meio da relação de número 10)

4.2.18. Identificador Sintético: Entidade Centro

O identificador sintético do centro é um número composto formado pelo *CEP* (sem máscaras) e pelo número do lote do centro. O identificador sintético foi utilizado para reduzir o custo de memória, já que a chave primária do centro é um valor que se repete várias vezes, em todas as salas. O identificador é composto dessa maneira, pois ele permite saber qual a localização de uma sala (*Endereço*) sem precisar realizar a operação de busca para encontrar o *Centro*.

4.2.19. Identificador Sintético: Entidade Sala

O identificador sintético da sala é um número composto formado pelo identificador do centro, por um número entre 0 e 3, que representa o tipo de sala, pelo número do bloco e pelo número da sala. O identificador sintético foi utilizado para reduzir o custo de memória, já que a chave primária da sala é um valor que se repete várias vezes, em todas as relações que envolvem salas. O identificador é composto dessa maneira, pois ele permite saber qual a localização de uma sala e qual o tipo de sala, que são valores que são importantes para todas as operações envolvendo salas, sem precisar realizar a operação de busca pela sala para encontrar o centro ao qual ela pertence.

4.2.20. Atributo derivado: Nº de Docentes

Solução adotada: O atributo será salvo na base de dados e será modificado sempre que um docente for adicionado ou removido de um centro.

Vantagens: Maior performance, pois não será necessário recalcular o número de docentes toda vez que uma consulta é realizada.

Desvantagens: Gasto de memória, dado que, para cada centro, será necessário armazenar um número a mais.

Solução alternativa: Não salvar este número e calculá-lo sempre que ele for ser utilizado.

4.2.21. Atributo derivado: Nº de Matriculados

Solução adotada: O atributo será salvo na base de dados e será modificado sempre que um aluno for matriculado ou sair de uma disciplina.

Vantagens: Maior performance, pois não será necessário recalcular o número de docentes toda vez que uma consulta é realizada.

Desvantagens: Gasto de memória, dado que, para cada disciplina, será necessário armazenar um número a mais.

Solução alternativa: Não salvar este número e calculá-lo sempre que ele for ser utilizado.

5. Mudanças relacionadas à segunda entrega

- Foram adicionadas as descrições para os atributos derivados *Nº de Docentes* e *Nº de Matriculados*.
- Algumas das descrições foram explicadas de modo mais detalhado, como a questão de performance, soluções alternativas, vantagens, desvantagens e justificativas de uso. Foram modificadas as entidades *Pessoa* e *Sala* e os relacionamentos *Contém* e *Inscreve*.
- Correção da chave estrangeira de *Manutenção*, que estava com o sentido da seta errado.

6. Terceira Entrega: Implementação

6.1. Consultas

6.1.1. Número médio de disciplinas ministradas por professor para cada um dos centros

```sql
SELECT C.CEP, C.NUMERO, ROUND(COALESCE(AVG(PA.NAULAS), 0),2)
FROM CENTRO C
LEFT JOIN (
    SELECT P.CPF, CO.CENTRO, COUNT(D.SIGLA) AS NAULAS
    FROM PROFESSOR P
    JOIN COORDENADOR CO ON P.SUPERVISOR = CO.CPF
    LEFT JOIN DISCIPLINA D ON D.PROFESSOR = P.CPF
    GROUP BY P.CPF, CO.CENTRO
) PA ON C.ID = PA.CENTRO
GROUP BY C.CEP, C.NUMERO;
```

Essa consulta em SQL retorna informações sobre os centros, incluindo o CEP, número e média de aulas ministradas por professores em cada centro. A consulta utiliza um LEFT JOIN com uma subconsulta que calcula o número de aulas ministradas por cada professor em cada centro. Os resultados são agrupados pelo CEP e número do centro, e a média das aulas é calculada usando a função AVG. Caso um centro não tenha professores associados, a média será 0.

6.1.2. Número de eventos de cada proposta em cada região (CEP)

```sql
SELECT EQ.CEP, P.NOME, COUNT(*) AS QUANTIDADE_DE_EVENTOS
FROM PROPOSTA P
JOIN (
    SELECT SC.CEP, E.PROPOSTA
    FROM (
        SELECT PROPOSTA, SALA, HORARIO
        FROM EVENTO
        WHERE BOOLABERTO = 'PUBLICO'
    ) E
    JOIN (
        SELECT S.ID, C.CEP
        FROM SALA S
        JOIN CENTRO C ON S.CENTRO = C.ID
    ) SC ON SC.ID = E.SALA
) EQ ON EQ.PROPOSTA = P.ID
GROUP BY EQ.CEP, P.NOME
ORDER BY EQ.CEP, P.NOME DESC;
```

A consulta em SQL busca obter informações sobre eventos públicos relacionados a propostas, agrupando-os por CEP (Código de Endereçamento Postal) e nome da proposta. A consulta envolve junções de subconsultas para obter os dados necessários. Os eventos públicos são selecionados com base na condição BOOLABERTO = 'PUBLICO'. As informações das salas e seus respectivos CEPs são obtidas através de uma junção entre as tabelas SALA e CENTRO. A tabela temporária resultante dessas junções é então unida à tabela PROPOSTA para relacionar as propostas aos eventos. Os resultados são agrupados pelo CEP e nome do proponente e ordenados pelo CEP em ordem crescente e pelo nome em ordem decrescente. O resultado final é uma lista de CEPs, nomes das propostas e a quantidade de eventos associados a cada combinação de CEP e nome.

6.1.3. CPFs de todos os técnicos que já realizaram manutenções em todos os centros

```sql
SELECT T.CPF
FROM TECNICO T
WHERE NOT EXISTS(
    SELECT ID FROM CENTRO
    EXCEPT
    SELECT DISTINCT S.CENTRO AS ID
    FROM SALA S, (
        SELECT DISTINCT ES.SALA
        FROM (
```

```
            SELECT EQUIPAMENTO
            FROM MANUTENCAO M
            WHERE T.CPF = M.TECNICO
        ) MT
        JOIN EQUIPAMENTOSALA ES ON ES.EQUIPAMENTO = MT.EQUIPAMENTO
    ) ST
    WHERE S.ID = ST.SALA
);
```

Este código SQL é utilizado para obter o CPF dos técnicos que estão associados a todos os centros através das salas onde realizaram manutenção em equipamentos específicos. A consulta principal utiliza a cláusula SELECT para selecionar o campo CPF da tabela TECNICO. A cláusula FROM especifica a tabela TECNICO como a fonte de dados. A cláusula WHERE é usada para filtrar os resultados, verificando se não existe uma determinada condição na subconsulta interna. A subconsulta interna envolve a tabela CENTRO e as tabelas SALA, MANUTENCAO e EQUIPAMENTOSALA, selecionando as salas onde os equipamentos das manutenções estão presentes. Em resumo, o código verifica se não há centros ausentes da lista de centros associados às salas onde os técnicos realizaram manutenção, e seleciona o CPF dos técnicos que estão associados a todos os centros.

6.1.4. Nome dos coordenadores que supervisionam mais de 3 professores

```
SELECT P.NOME
FROM COORDENADOR C
JOIN PESSOA P ON C.CPF = P.CPF
JOIN(
      SELECT SUPERVISOR, COUNT(*)
      FROM PROFESSOR
      GROUP BY SUPERVISOR
      HAVING COUNT(*) > 3
) D ON P.CPF = D.SUPERVISOR;
```

O código SQL seleciona o nome dos coordenadores que atuam como supervisores de mais de três professores. Isso é alcançado por meio de uma subconsulta que agrupa os professores por supervisor e filtra apenas aqueles com mais de três professores sob sua supervisão. A consulta final junta a tabela de coordenadores com a subconsulta, obtendo o nome das pessoas que correspondem aos coordenadores que atendem aos critérios estabelecidos.

6.1.5. Ranqueia os assuntos das disciplinas (representados pelo primeiro caractere da sigla)

```
SELECT DISTINCT SUBSTR(D.SIGLA, 1, 1) AS TOPICO,
                SUM(HD.NHORARIOS) AS NHORARIOS,
                RANK() OVER (ORDER BY SUM(HD.NHORARIOS) DESC) AS RANK_NUMERO_AULAS
FROM DISCIPLINA D
JOIN (
    SELECT DISC, COUNT(*) AS NHORARIOS
    FROM HORARIOSDISC
    GROUP BY DISC
) HD ON D.SIGLA = HD.DISC
GROUP BY SUBSTR(D.SIGLA, 1, 1);
```

O código SQL apresentado realiza uma consulta em um banco de dados para obter informações sobre disciplinas. Ele agrupa as disciplinas por tópico (representado pelo primeiro caractere da sigla) e calcula a soma do número de horários para cada disciplina. Em seguida, classifica os tópicos com base na soma do número de horários, atribuindo um número de classificação a cada grupo. O resultado é uma tabela com as colunas "TOPICO" (tópico da disciplina), "NHORARIOS" (soma do número de horários) e "RANK_NUMERO_AULAS" (classificação dos tópicos).

6.2. Aplicação

6.3. Descrição Inicial

Como aplicativo foi feito um shell interativo em python, utilizando a biblioteca Psycopg2 que faz a comunicação entre P*ython*(cliente) e o *PostgreSQL*(servidor).

6.4. Requisitos do sistema

O sistema foi implementado utilizando o sistema operacional linux, para seu funcionamento é necessário que esteja instalada as seguinte linguagens/bibliotecas:

- Python 3
- PostgreSql
- Psycopg2

6.5. Introdução para utilização

Utiliza o comando para rodar o aplicativo:

```
>python3 app.py
```

Em seguida a seguinte tela será mostrada:

```
conectar-se ao banco postgree
(1) usuário padrão
(2) Outro usuário
(0) Sair
:
```

o usuário padrão tem os seguintes dados:

```
banco_nome = "projeto"
usuario = "postgres"
senha = "123"
```

Caso não seja o usuário desejado a segunda opção irá pedir os dados para o usuário que se deseja conectar.

Logo em seguida a seguinte tela aparecerá:

```
      Connecting to the PostgreSQL database...
PostgreSQL database version:
('PostgreSQL 14.8 (Ubuntu 14.8-0ubuntu0.22.04.1) on
x86_64-pc-linux-gnu, compiled by gcc (Ubuntu
11.3.0-1ubuntu1~22.04.1) 11.3.0, 64-bit',)


Aplicativo de acesso ao banco de dados


(1) Buscar dados no sistema
(2) Inserir novo dado no sistema
(0) Sair
```

A tela mostra os dados sobre as versões utilizadas no aplicativo, e em seguida uma lista de comando.

5.1. Inserção Implementada

Para a funcionalidade de inserção foi escolhido a adição de uma nova proposta de evento nos dados recebidos é conferido se as strings recebidas são de um tamanho válido.

```
def inserir_proposta(conn, cur):
    sql = """INSERT INTO PROPOSTA VALUES (%s, %s, %s);"""
    valores = []
```

```
    valores.append(int(input("id:")))
    valores.append(input("nome:"))
    valores.append(input("descrição:"))
    try:
        cur.execute(sql, (valores))

        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
```

5.2. Busca Implementada

Para a funcionalidade de busca um protótipo de uma função genérica foi feito, no momento ela acessa apenas a tabela pessoa e pergunta ao usuário quais colunas ele gostaria de ver.

```
def busca(cur):
    try:
        clear()
        print("(1) Tabela pessoas")
        print("(0) Voltar")
        escolha = entrada(':',['1','0'])

        if escolha == '0':
            return
        elif escolha == '1':
            val = []
            if entrada("printar todos os valores ?(S/N)",['S','N']) == 'S':
                val.append("*")
            else:
                if entrada("printar CPf?(S/N)",['S','N']) == 'S':
                    val.append("CPF")
                if entrada("printar Nome?(S/N)",['S','N']) == 'S':
                    val.append("NOME")
                if entrada("printar tipo?(S/N)",['S','N']) == 'S':
                    val.append("tipo")
                if entrada("printar telefone?(S/N)",['S','N']) == 'S':
                    val.append("telefone")
                if entrada("printar nascimento?(S/N)",['S','N']) == 'S':
                    val.append("nascimento")
                if entrada("printar sexo?(S/N)",['S','N']) == 'S':
                    val.append("sexo")
            val = ",".join(val)
            sql = f"select {val} from pessoa"
```

```
            cur.execute(sql)

        resultado = cur.fetchall()
        print()
        for res in resultado:
            print(res)
    except (Exception, psycopg2.DatabaseError) as error:
        print("")
        print(error)
```

7. Conclusão

O grupo 10 vivenciou um grande percurso ao decorrer da matéria de Banco de Dados, visto que desfrutamos de tudo que a disciplina tinha a oferecer, como as monitorias, as reuniões e as aulas. As monitorias que tivemos foram cruciais para esclarecermos dúvidas, que poderiam impedir o desenvolvimento de um trabalho de qualidade consistente.

Este trabalho permitiu ao nosso grupo observar, de forma empírica, onde poderemos utilizar cada parte da matéria que nos foi ensinada. Assim, seremos capazes de utilizá-la com proficiência no futuro.

Durante a disciplina de Banco de Dados, enfrentamos um desafio significativo na 1° entrega do trabalho. Foi exigido um esforço adicional para compreender a estrutura e as conexões do sistema em desenvolvimento. Uma das dificuldades encontradas foi a tradução dos elementos do mundo real para o contexto de um banco de dados, uma tarefa que exigiu considerar tanto a natureza objetiva do banco de dados quanto a subjetividade na abstração dos conceitos da realidade. Diante disso, exploramos diversas alternativas para solucionar cada problema apresentado.

A possibilidade de podermos modificar as nossas entregas anteriores, juntamente com os feedbacks do monitor, nos proporcionou uma visão detalhada do processo de desenvolvimento de um banco de dados e suas nuances, assim como uma forma de revisarmos e aprendermos com os erros cometidos previamente.

Em conclusão, acreditamos que houve um aprendizado relevante, e estamos gratos pela experiência.