

”

**E-fólio A** | Folha de resolução para E-fólio

**UNIDADE CURRICULAR:** Fundamentos de Bases de Dados

**CÓDIGO:** 21053

**DOCENTE:** Paulo Pombinho

**A preencher pelo estudante**

**NOME:** Vítor Manuel Metrogos Frango

**N.º DE ESTUDANTE:** 1802925

**CURSO:** Licenciatura em Engenharia Informática

**DATA DE ENTREGA:** 16 de novembro 2023

## TRABALHO / RESOLUÇÃO:

**1) A consistência em transações** de base de dados é a garantia de que cada transação mantém a precisão dos dados, ou seja, uma transação deve transformar a base de dados de um estado consistente para outro. Essa consistência é essencial para preservar a integridade dos dados e prevenir erros ou problemas que possam surgir de dados imprecisos ou inconsistentes.

A consistência é um dos princípios fundamentais do modelo ACID que descreve um conjunto de quatro propriedades-chave que definem uma transação: Atomicidade, Consistência, Isolamento e Durabilidade, se uma base de dados atende a estas quatro propriedades, ela é considerado compatível com ACID. Isso é especialmente crucial para garantir que todas as transações sejam processadas de maneira confiável.

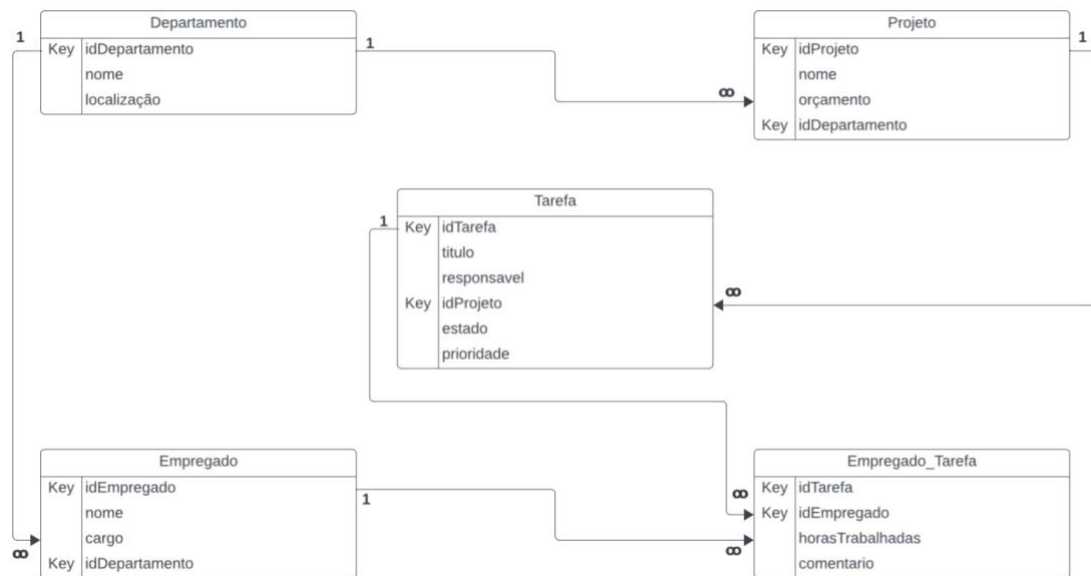
Se um sistema de base de dados ACID não garantir consistência, podem, como é logico **surgir diversas inconsistências** que comprometem a integridade dos dados. Alguns exemplos destas inconsistências que podem ocorrer são:

1. **Violação de Integridade Referencial:** A falta de consistência pode causar violações na integridade referencial, onde as relações entre as tabelas não são mantidas corretamente. Uma transação mal executada pode resultar em chaves estrangeiras que não correspondem às chaves primárias em outras tabelas.
2. **Dados Órfãos ou Perdidos:** Se uma transação falhar durante o processo, pode resultar na criação de dados órfãos (dados que não têm relação com outras partes da base de dados) ou na perda de dados que deveriam ter sido inseridos ou atualizados.
3. **Inconsistências de Valor:** Transações concorrentes sem isolamento adequado podem resultar em leituras ou gravações de valores inconsistentes. Isso pode ocorrer quando uma transação lê dados que estão a ser modificados por outra transação simultaneamente.
4. **Estados Inconsistentes da base de Dados:** A falta de atomicidade pode resultar em estados inconsistentes da base de dados, onde algumas partes da transação foram aplicadas e outras não. Isso pode deixar a base de dados num estado indesejado e difícil de recuperar.
5. **Inconsistências de Restrição:** Se as restrições da base de dados não forem aplicadas corretamente, os dados podem violar as regras de integridade definidas, resultando em inconsistências em níveis lógicos e semânticos.
6. **Problemas de Concorrência:** Sem isolamento adequado, transações simultâneas podem interferir umas nas outras, resultando em resultados imprevisíveis e inconsistências nos dados.
7. **Falhas de Durabilidade:** Se a durabilidade não for garantida, as atualizações feitas durante uma transação podem ser perdidas em caso de falha do sistema, comprometendo a persistência e a confiabilidade dos dados.

Em suma, a consistência no contexto de transações ACID é crucial para garantir que a base de dados permaneça num estado válido e íntegro, mesmo diante de falhas, transações concorrentes e outros eventos imprevistos.

Por fim, considera se como exemplo, um sistema bancário onde uma transação movimenta dinheiro de uma conta para outra, se por um motivo externo a transação for interrompida após o dinheiro ser debitado da primeira conta, mas antes de ser creditado na segunda, o dinheiro “desaparecerá”. Isso resultará numa inconsistência de dados, pois o total de dinheiro no sistema bancário terá diminuído de forma incorreta. Portanto, é de grande importância que os SGBDs garantam a consistência de todas as transações para evitar tais problemas.

2)



Tabelas / Atributos / Relações / chaves primarias e estrangeiras

**Tabela Departamento** -> Atributos: idDepartamento (chave primária), nome, localização.

**Tabela Projeto** -> Atributos: idProjeto (chave primária), nome, orçamento, idDepartamento (chave estrangeira referenciando idDepartamento da tabela Departamento).

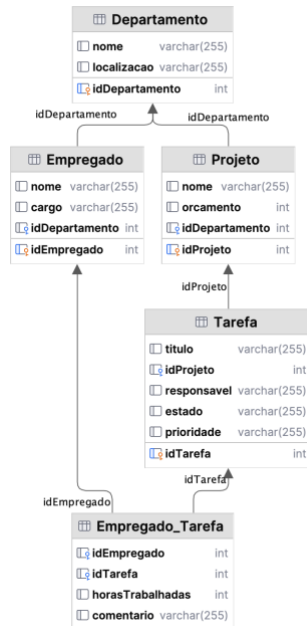
**Tabela Tarefa** -> Atributos: idTarefa (chave primária), título, responsável, idProjeto (chave estrangeira referenciando idProjeto da tabela Projeto), estado, prioridade.

**Tabela Empregado** -> Atributos: idEmpregado (chave primária), nome, cargo, idDepartamento (chave estrangeira referenciando idDepartamento da tabela Departamento).

**Tabela Empregado\_Tarefa** -> Atributos: idTarefa (chave estrangeira referenciando idTarefa da tabela Tarefa), idEmpregado (chave estrangeira referenciando idEmpregado da tabela Empregado), horasTrabalhadas, comentário.

## Criação de tabelas atributos e relações em SQL

```
CREATE TABLE Departamento (  
    idDepartamento INT PRIMARY KEY,  
    nome VARCHAR(255),  
    localizacao VARCHAR(255)  
);  
  
CREATE table Projeto  
(  
    idProjeto INT PRIMARY KEY,  
    nome VARCHAR(255),  
    orcamento INT,  
    idDepartamento INT,  
    FOREIGN KEY (idDepartamento) REFERENCES  
Departamento(idDepartamento)  
);  
  
create table Tarefa  
(  
    idTarefa INT PRIMARY KEY,  
    titulo VARCHAR(255),  
    idProjeto INT,  
    responsavel VARCHAR(255),  
    estado VARCHAR(255),  
    prioridade VARCHAR(255),  
    FOREIGN KEY (idProjeto) REFERENCES Projeto(idProjeto)  
);  
  
create table Empregado  
(  
    idEmpregado INT PRIMARY KEY,  
    nome VARCHAR(255),  
    cargo VARCHAR(255),  
    idDepartamento INT,  
    FOREIGN KEY (idDepartamento) REFERENCES  
Departamento(idDepartamento)  
);  
  
create table Empregado_Tarefa  
(  
    idEmpregado INT,  
    idTarefa INT,  
    horasTrabalhadas INT,  
    comentario VARCHAR(255),  
    FOREIGN KEY (idEmpregado) REFERENCES Empregado(idEmpregado),  
    FOREIGN KEY (idTarefa) REFERENCES Tarefa(idTarefa)  
);
```



## Introdução de dados

```

INSERT INTO Departamento (idDepartamento, nome, localizacao)
VALUES (1, 'Departamento A', 'Local A'),
       (2, 'Departamento B', 'Local B'),
       (3, 'Departamento C', 'Local C');
  
```

```

INSERT INTO Projeto (idProjeto, nome, orcamento, idDepartamento)
VALUES (1, 'Projeto X', 10000.00, 1),
       (2, 'Projeto Y', 15000.50, 2),
       (3, 'Projeto Z', 12000.75, 3);
  
```

```

INSERT INTO Tarefa (idTarefa, titulo, responsavel, idProjeto, estado,
prioridade)
VALUES (1, 'Tarefa 1', 'Responsavel 1', 1, 'Em Andamento', 'Alta'
       (2, 'Tarefa 2', 'Responsavel 2', 2, 'Concluída', 'Baixa'),
       (3, 'Tarefa 3', 'Responsavel 3', 3, 'Pendente', 'Alta')
       (3, 'Tarefa 3', 'Responsavel 3', 3, 'Pendente', 'Normal');
  
```

```

INSERT INTO Empregado (idEmpregado, nome, cargo, idDepartamento)
VALUES (1, 'Empregado 1', 'Cargo 1', 1),
       (2, 'Empregado 2', 'Cargo 2', 2),
       (3, 'Empregado 3', 'Cargo 3', 3);
  
```

```

INSERT INTO Empregado_Tarefa (idTarefa, idEmpregado, horasTrabalhadas,
comentario)
VALUES (1, 1, 8, 'Concluído no prazo'),
       (2, 2, 10, 'Tarefa complexa'),
       (3, 3, 6, 'Atraso devido a problemas técnicos');
  
```

3a)

The screenshot shows a SQL IDE with a query editor and an output window. The query editor contains the following SQL code:

```

VALUES (1, 1, 8, 'Concluído no prazo'),
(2, 2, 10, 'Tarefa complexa'),
(3, 3, 6, 'Atraso devido a problemas técnicos');

SELECT
    Projeto.nome AS NomeProjeto,
    Tarefa.estado,
    Tarefa.prioridade,
    Departamento.nome AS NomeDepartamento
FROM
    Projeto
JOIN
    Tarefa ON Projeto.idProjeto = Tarefa.idProjeto
JOIN
    Departamento ON Projeto.idDepartamento = Departamento.idDepartamento
WHERE
    Tarefa.estado = 'Completo' AND Tarefa.prioridade = 'Alta';

```

The output window shows the result of the query, which is a single row:

NomeProjeto	estado	prioridade	NomeDepartamento
Projeto Y	Completo	Alta	Finanças

3b)

The screenshot shows a SQL IDE with a query editor and an output window. The query editor contains the following SQL code:

```

Tarefa.estado = 'Completo' AND Tarefa.prioridade = 'Alta';

SELECT
    Empregado.nome AS NomeEmpregado,
    SUM(Empregado_Tarefa.horasTrabalhadas) AS TotalHoras,
    Departamento.nome AS NomeDepartamento
FROM
    Empregado
JOIN
    Empregado_Tarefa ON Empregado.idEmpregado = Empregado_Tarefa.idEmpregado
JOIN
    Tarefa ON Empregado_Tarefa.idTarefa = Tarefa.idTarefa
JOIN
    Departamento ON Empregado.idDepartamento = Departamento.idDepartamento
GROUP BY
    Empregado.idEmpregado, NomeDepartamento
ORDER BY
    TotalHoras DESC;

```


The output window shows the result of the query, which is a table with 3 rows:

NomeEmpregado	TotalHoras	NomeDepartamento
Empregado 1	18	Investigação
Empregado 2	10	Finanças
Empregado 3	6	Obras

**3c)**

```
202
203 ✓ SELECT Empregado.nome AS NomeEmpregado, COUNT(Empregado_Tarefa.idTarefa) AS NumTarefas
204 FROM Empregado
205 JOIN Empregado_Tarefa ON Empregado.idEmpregado = Empregado_Tarefa.idEmpregado
206 GROUP BY Empregado.idEmpregado
207 HAVING COUNT(Empregado_Tarefa.idTarefa) > (
208     SELECT AVG(TarefasPorEmpregado.NumTarefas) AS MediaTarefasPorEmpregado
209     FROM (
210         SELECT COUNT(idTarefa) AS NumTarefas
211         FROM Empregado_Tarefa
212         GROUP BY idEmpregado
213     ) AS TarefasPorEmpregado
214 )
215
216
```

**3d)**



The screenshot shows a SQL editor interface with a toolbar at the top containing icons for running queries, saving, and other functions. The editor displays a SQL query to update the 'prioridade' field in the 'Tarefa' table. The query is as follows:

```

UPDATE Tarefa
SET prioridade = 'Urgente'
WHERE idProjeto IN (
    SELECT idProjeto
    FROM Projeto
    WHERE idDepartamento IN (SELECT idDepartamento
    FROM Departamento
    WHERE nome = 'Finanças'
    )
)
AND estado = 'Em Progresso'
AND idTarefa IN (
    SELECT idTarefa
    FROM Empregado_Tarefa
    WHERE horasTrabalhadas > 10
);

```

The query is highlighted with a blue selection box. The editor also shows line numbers on the left side, ranging from 183 to 201.

## Bibliografia

(1) Database System Concepts, Henry F. Korth, Abraham Silberschatz, and S. Sudarshan

(2) "What are ACID Transactions?" - Databricks.  
<https://www.databricks.com/glossary/acid-transactions>.

(3) "ACID Model vs BASE Model For Database" - GeeksforGeeks.  
<https://www.geeksforgeeks.org/acid-model-vs-base-model-for-database/>.

(4) <http://db-book.com/>

(5) The Theory of Relational Databases, 1983 David Maier, <http://web.cecs.pdx.edu/~maier/TheoryBook/TRD.html>

(6) copilot for DataGrip (JetBrains)