



SQL: a construção de um banco de dados relacional

Podemos aprender com as abelhas:

- Elas apresentam uma riqueza comportamental em sua sociedade e uma impressionante capacidade cognitiva que cada vez mais têm fascinado pesquisadores.

Apresentarei na linguagem e na forma que compreendi, de forma simples e prática.

Nesta jornada de estudos, que encaixo as informações como um quebra-cabeças .

Vou agregando as informações e faço conexões com os meus conhecimentos pré-existentes.

Segundo as teorias de aprendizagem através da perspectiva de David Ausubel (Ausubel; Novak; Hanesian, 1980; Ausubel, 2003.), a ocorrência de aprendizagem significativa depende de uma estratégia que possibilite ao aluno vincular os novos conhecimentos a serem aprendidos com conhecimentos existentes em sua estrutura cognitiva.

Dessa forma, é necessário que haja conhecimentos básicos para que o aluno possa agregar cada novo conhecimento a ser aprendido.

SQL definições e principais funções:

O **SQL** é uma linguagem relacional, isto significa que ela conecta os dados entre as diferentes tabelas (estas tabelas são semelhantes às linhas e colunas no Excel).

Você faz consultas, que são chamadas de “query” e obtêm dados, também pode adicionar e modificar dados.

Os dados precisam ser bem estruturados e seguros, através de boas práticas.

Eu fiz download no “**SQLite** database engine” porque me pareceu “bem simpático” e prático aos meus objetivos de aprendizagem.

A seguir trarei as principais funcionalidades que aprendi. E um projeto que preparei a data set.

Nome	Tipo	Esquema
▼ Tabelas (6)		
▶ Equipamentos	CREATE TABLE "Equipamentos" ("ID" INTEGER NOT NULL, "Nome_Equipamento" TEXT NOT NULL, "Tipo" TEXT NOT NULL, "Quantidade" INTEGER NOT NULL, "Área" TEXT NOT NULL, "Data_compra" TEXT NOT NULL, "Pagamento_Mensal" NUMERIC NOT NULL, "CustoMensalOperação" NUMERIC NOT NULL, "Data_compra" TEXT NOT NULL, "Pagamento_Mensal" NUMERIC NOT NULL, "CustoMensalOperação" NUMERIC NOT NULL)	
▶ Equipamentos_Fornecedores	CREATE TABLE "Equipamentos_Fornecedores" ("Dias_Entrega" INTEGER NOT NULL, "QTDE Pedido" INTEGER NOT NULL, "ID" INTEGER NOT NULL, FOREIGN KEY("ID") REFERENCES "Equipamentos" ("ID"), "Nome_Fornecedor" TEXT NOT NULL, "Endereço" TEXT NOT NULL, "E-mail" TEXT NOT NULL, "Data_compra" TEXT NOT NULL, "Pagamento_Mensal" NUMERIC NOT NULL, "CustoMensalOperação" NUMERIC NOT NULL)	
▶ Estoques	CREATE TABLE "Estoques" ("Num_item" INTEGER NOT NULL, "Pedido" INTEGER NOT NULL, "Local" TEXT NOT NULL, "CustoUnidade" NUMERIC NOT NULL, "QTDE" INTEGER NOT NULL, "Data_compra" TEXT NOT NULL, "Pagamento_Mensal" NUMERIC NOT NULL, "CustoMensalOperação" NUMERIC NOT NULL)	
▶ Fornecedores	CREATE TABLE "Fornecedores" ("ID" INTEGER NOT NULL, "Nome" TEXT NOT NULL, "Email" TEXT NOT NULL, "Endereço" TEXT NOT NULL, "Escritório" TEXT NOT NULL, "Data_compra" TEXT NOT NULL, "Pagamento_Mensal" NUMERIC NOT NULL, "CustoMensalOperação" NUMERIC NOT NULL)	
▶ Funcionários	CREATE TABLE "Funcionários" ("id_Func" INTEGER PRIMARY KEY, "nome" TEXT NOT NULL, "Salario" NUMERIC (10,2))	
▶ sqlite_sequence	CREATE TABLE sqlite_sequence(name,seq)	
Índices (0)		
▼ Vistas (6)		
▶ CASE	CREATE VIEW "CASE" AS SELECT Nome, Email, CASE CustoMensalOperação WHEN CustoMensalOperação > 1000 THEN 'Custo alto apresentar detalhamento e notas fiscais' WHEN CustoMensalOperação < 1000 THEN 'Custo baixo apresentar detalhamento e notas fiscais' END AS CustoMensalOperação FROM Funcionários	
▶ Equip.	CREATE VIEW "Equip." AS SELECT * FROM "main"."Equipamentos" WHERE "ID" LIKE '%000001%' ESCAPE '\ ' AND "Nome_Equipamento" LIKE '%maquina de cafe%' ESCAPE '\ ' AND "Data_compra" LIKE '%2023-01-01%' ESCAPE '\ '	
▶ Union_all	CREATE VIEW "Union_all" AS SELECT sum(Salario)AS Total FROM Funcionários UNION ALL SELECT sum(Pagamento_Mensal)AS total FROM Fornecedores	
▶ distinct	CREATE VIEW "distinct" AS SELECT DISTINCT(Responsavel) FROM Equipamentos	
▶ equip	CREATE VIEW "equip" AS SELECT * FROM "main"."Equipamentos" WHERE "ID" LIKE '%000001%' ESCAPE '\ ' AND "Nome_Equipamento" LIKE '%maquina de cafe%' ESCAPE '\ ' AND "Data_compra" LIKE '%2023-01-01%' ESCAPE '\ '	
▶ null	CREATE VIEW "null" AS SELECT "ID","Nome","Email","Endereço","Escritório",strftime('%d/%m/%Y',"Data_compra") AS "Data_compra","Pagamento_Mensal","CustoMensalOperação" FROM Equipamentos	
▶ Gatilhos (0)		

NULL

Tipo de dado atualmente na célula: NULL
0 byte


Aplicar

CRIANDO TABELAS:

Antes de criar tabelas, é bom ter em mente os conceitos de chaves primárias, também conhecidas como PK e chaves estrangeiras.

Chaves primárias ou PK, são dados únicos, exclusivos contidos nas linhas da tabela relacional. Como exemplo, temos o CPF, o chassi do carro, número da conta bancária, etc. A PK garante a integridade dos dados.

No exemplo do meu banco de dados Suply_Chain, o id_Func é uma das chaves primárias (observe uma chave amarela na tabela e descrição das colunas a seguir:



Data_Contrato	NUMERIC	Data_Contrato NUMERIC NOT NULL
▼ Funcionários		CREATE TABLE Funcionários(id_Func INTEGER PRIMARY KEY, name TEXT, Salario NUMERIC (10,2))
id_Func	INTEGER	"id_Func" INTEGER
name	TEXT	"name" TEXT
Salario	NUMERIC(10, 2)	"Salario" NUMERIC(10, 2)

CRIANDO TABELAS:

Chaves estrangeiras.

As chaves estrangeiras são colunas, que representam os dados que pertencem a uma chave primária de outra tabela. Not Null, significa que o preenchimento destes dados são obrigatórios.



Equipamentos_Forn...	CREATE TABLE "Equipamentos_Fornecedores" ("Dias_Entrega " INTEGER NOT NULL, "QTDE Pedido" INTEGER NOT NULL, "ID" INTEGER NOT NULL, FOREIGN KEY("ID") REFERENCES "Fornecedores"("ID"), FOREIGN KEY("QTDE Pedido") REFERENCES "Equipamentos")
Dias_Entrega	INTEGER "Dias_Entrega " INTEGER NOT NULL
QTDE Pedido	INTEGER "QTDE Pedido" INTEGER NOT NULL
ID	INTEGER "ID" INTEGER NOT NULL

▼ Fornecedores	CREATE TABLE "Fornecedores" ("ID" INTEGER NOT NULL, "Nome" TEXT NOT NULL, "Email" TEXT NOT NULL, "Endereço" TEXT NOT NULL, "Escritorio" TEXT NOT NULL, "Data_con
ID	INTEGER "ID" INTEGER NOT NULL

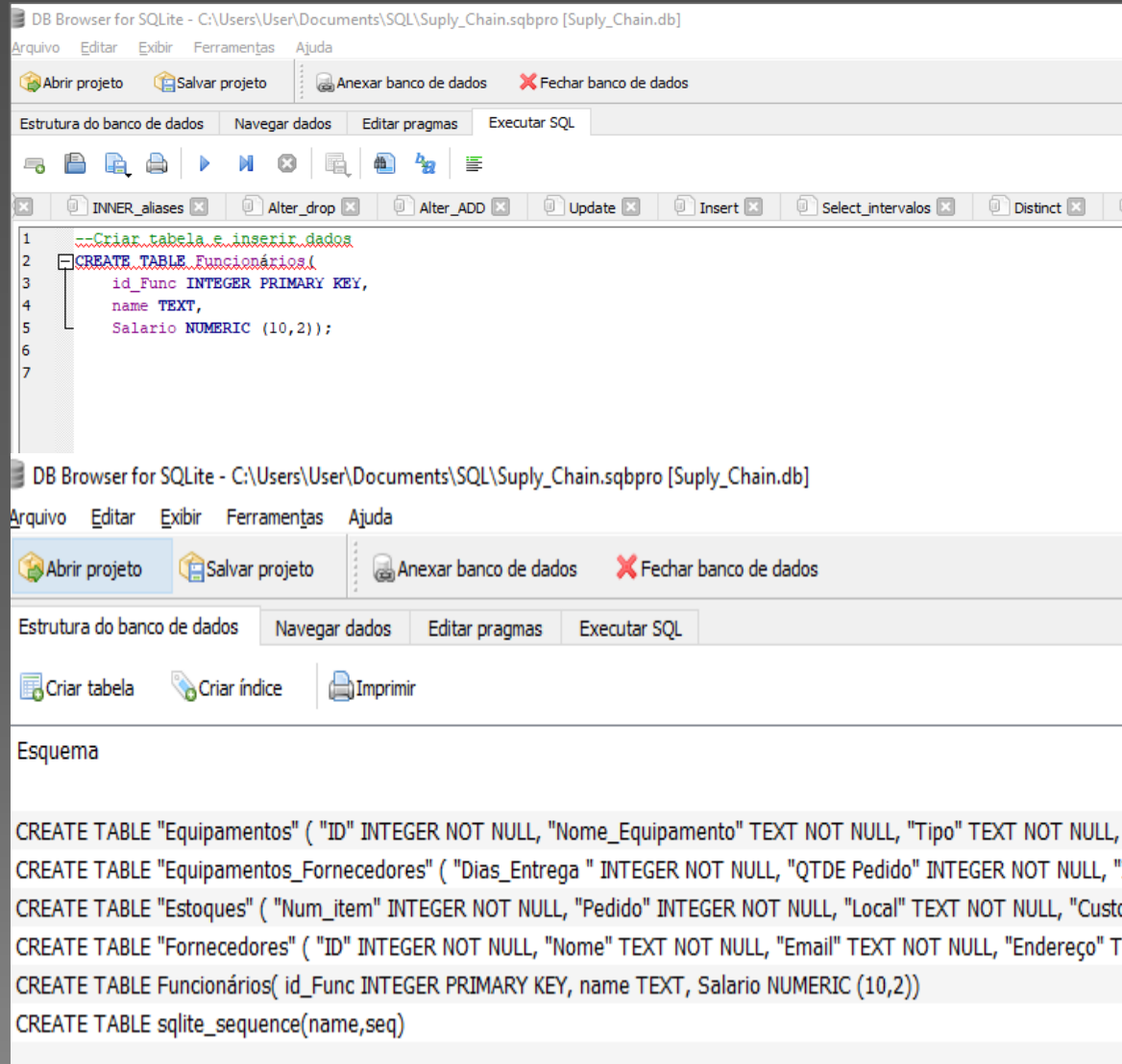
criação de tabelas: script ou direto na estrutura de dados

A tabela pode ser criada com comandos na aba Executar SQL, criar aba:

CREATE TABLE "nome da tabela".

Colocar entre parênteses as colunas a serem criadas, definindo os valores das classes, integer, numeric, text. Há outras definições como char, que é o tamanho do texto. Finalizando com ponto e vírgula.

O método mais fácil é criar a tabela na estrutura do banco de dados (2ª imagem) você já escolhe as opções em uma tabela.



Para criar a query,
Clicar em executar
SQL e em seguida
criar aba, conforme
imagem a seguir:



Principais
comandos nesta
query:

- Select
- Sum
- From

Custo unidade =
coluna na tabela
Estoques

CRIANDO QUERIES:

DB Browser for SQLite - C:\Users\User\Documents\SQL\Suply_Chain.sqbpro [Suply_Chain.db]

Arquivo Editar Exibir Ferramentas Ajuda

Abrir projeto Salvar projeto Anexar banco de dados Fechar banco de dados

Estrutura do banco de dados Navegar dados Editar pragmas Executar SQL

Query1 Query2_Count Where_clause Queries.sql INNER_aliases

```
1  -- Primeira query sobre aliens
2
3  SELECT
4  sum(CustoUnidade) AS sum_CustoUnidade
5  FROM Estoques
6
```

	sum_CustoUnidade
1	7791

Execução finalizada sem erros.
Resultado: 1 linhas retornadas em 8 ms
Na linha 1:
-- Primeira query sobre aliens

```
SELECT
sum(CustoUnidade) AS sum_CustoUnidade
FROM Estoques
```


PARA CONSULTAR O OBJETO DESEJADO, UTILIZAR WHERE CLAUSE

DB Browser for SQLite - C:\Users\User\Documents\SQL\Suply_Chain.sqbpro [Suply_Chain.db]

Arquivo Editar Exibir Ferramentas Ajuda

Abrir projeto Salvar projeto Anexar banco de dados Fechar banco de dados

Estrutura do banco de dados Navegar dados Editar pragmas Executar SQL

Query1 Query2_Count Where_clause Queries.sql INNER

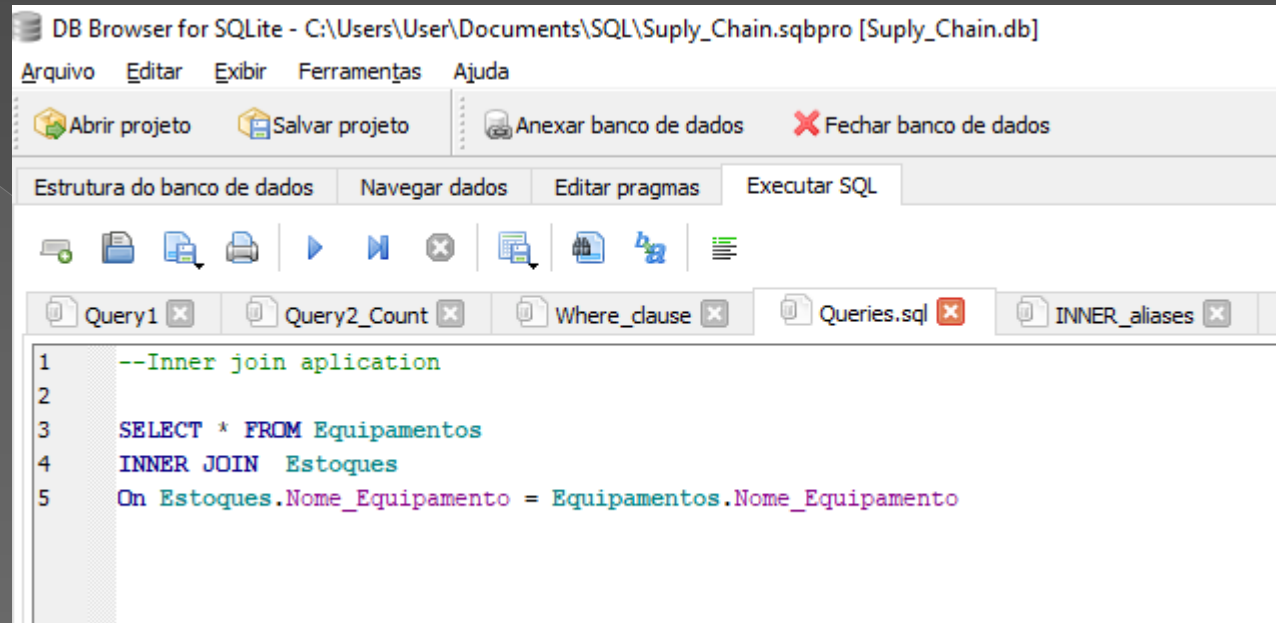
```
1 --Utilização de Where clause
2
3 SELECT SUM (CustoMensalOperação)
4 FROM Fornecedores
5 WHERE ID = 2
```

	SUM (CustoMensalOperação)
1	NULL

Execução finalizada sem erros.
Resultado: 1 linhas retornadas em 8 ms
Na linha 1:
--Utilização de Where clause

```
SELECT SUM (CustoMensalOperação)
FROM Fornecedores
WHERE ID = 2
```

O COMANDO INNER JOIN É MUITO UTILIZADO PARA COMPARAR DADOS EM MAIS DE UMA TABELA.



The screenshot shows the DB Browser for SQLite interface. The title bar indicates the database file is 'C:\Users\User\Documents\SQL\Suply_Chain.sqbpro [Suply_Chain.db]'. The menu bar includes 'Arquivo', 'Editar', 'Exibir', 'Ferramentas', and 'Ajuda'. The toolbar has buttons for 'Abrir projeto', 'Salvar projeto', 'Anexar banco de dados', and 'Fechar banco de dados'. Below the toolbar are tabs for 'Estrutura do banco de dados', 'Navegar dados', 'Editar pragmas', and 'Executar SQL'. The 'Executar SQL' tab is active, showing a query editor with the following SQL code:

```
1  --Inner join aplicacion
2
3  SELECT * FROM Equipamentos
4  INNER JOIN Estoques
5  On Estoques.Nome_Equipamento = Equipamentos.Nome_Equipamento
```

	ID	Nome_Equipamento	Tipo	Quantidade	Área	Ano Garantia	Num_item	Pedido	Local	CustoUnidade	QTDE	Valor_Total	Nome_Equipamento	Responsavel
1	4	maquina pequena cappucino	residencia	560	B	3	4	983	Curitiba	930	100	93000	maquina pequena cappucino	Jose
2	12	maquina cafe expresso	grãos	200	residencia	5	1	980	São Paulo	1080	200	216000	maquina cafe expresso	Karen
3	17	moedores	Industrial	230	empresas	8	7	985	Curitiba	831	10	8310	moedores	Jose

Execução finalizada sem erros.

Resultado: 3 linhas retornadas em 29 ms

Na linha 1:

--Inner join aplicacion

SELECT * FROM Equipamentos

INNER JOIN Estoques

On Estoques.Nome_Equipamento = Equipamentos.Nome_Equipamento

LEFT E RIGHT OUTER JOIN

DB Browser for SQLite - C:\Users\User\Documents\SQL\Suply_Chain.sqbpro [Suply_Chain.db]

Arquivo Editar Exibir Ferramentas Ajuda

Abrir projeto Salvar projeto Anexar banco de dados Fechar banco de dados

Estrutura do banco de dados Navegar dados Editar pragmas Executar SQL

Query1 Query2_Count Where_clause Queries.sql INNER_alias

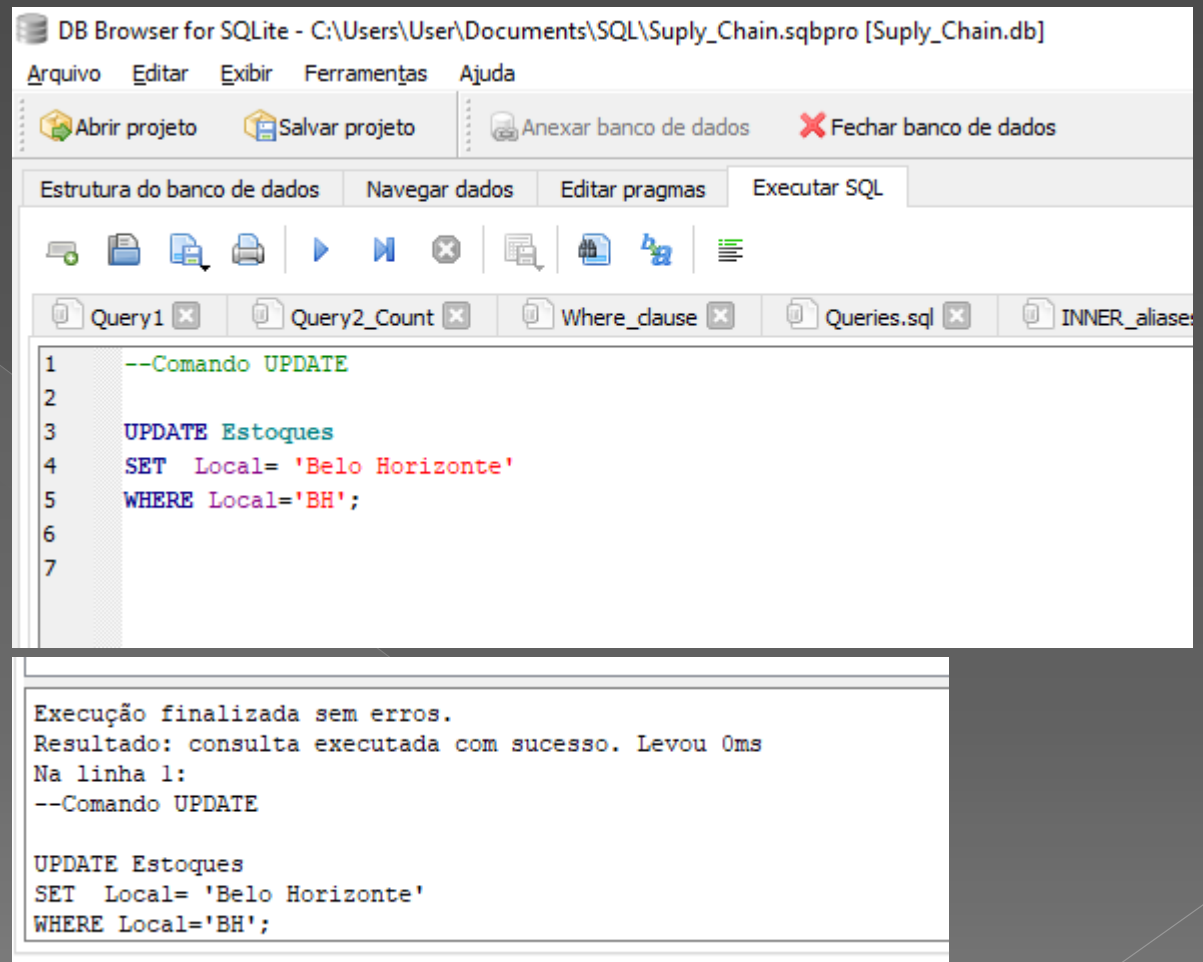
```
1 --Left Outer join NULL
2 SELECT * FROM Equipamentos
3 LEFT OUTER JOIN Fornecedores
4 ON Equipamentos.ID = Fornecedores.ID
5 WHERE Data_Contrato is NOT NULL
```

	ID	Nome_Equipamento	Tipo	Quantidade	Área	Ano Garantia	ID	Nome	Email	Endereço	Escritorio	Data_compra	Pagamento_Mensal	CustoMensalOperação	Data_Contrato
1	3	maquina café em graos	cafeteria	350	A	5	3	Clara	nnevcc@gmail.com	Avenida Getulio Vargas,23	BH	2023-11-12	1700	170	2018-03-09
2	4	maquina pequena cappuccino	residencia	560	B	3	4	Klever	kkferr@gmail.com	Av.Beira Mar, 2023	Fortaleza	2022-10-07	5000	650	2022-10-08
3	5	maquina automatica	predios empresariais	700	C	6	5	Joao	jatun@yomail.com	Av.Paulista, 1274	São Paulo	2023-12-12	10000	1008	2023-02-03
4	6	maquina café filtrado	predio empresarial	200	C	5	6	Manuel	klainman@goiçl.com	Rua Monções, 14	BH	2023-11-11	12000	1250	2023-11-10
5	7	moedores de café	cafeteria	180	A	3	7	Candida Soares		Av.Nações Unidas, 1800	São Paulo	2022-10-10	8000	830	2022-03-03
6	8	garrafas termicas	residencia	51	D	1	8	Danilo Pereira		Rua da Liberdade, 5023	São Paulo	2023-07-08	9000	950	2023-07-01

Execução finalizada sem erros.
Resultado: 9 linhas retornadas em 16 ms
Na linha 1:
--Left Outer join NULL
SELECT * FROM Equipamentos
LEFT OUTER JOIN Fornecedores
ON Equipamentos.ID = Fornecedores.ID
WHERE Data_Contrato is NOT NULL

ESTA QUERY JUNTA DUAS TABELAS, É UTIL PARA VISUALIZAR POSSÍVEIS DISCREPÂNCIAS NO BANCO DE DADOS.

COMANDO UPDATE



COMANDO INSERT INTO



The screenshot shows the DB Browser for SQLite application. The title bar indicates the file path: C:\Users\User\Documents\SQL\Suply_Chain.sqbpro [Suply_Chain.db]. The menu bar includes Arquivo, Editar, Exibir, Ferramentas, and Ajuda. The toolbar contains buttons for 'Abrir projeto', 'Salvar projeto', 'Anexar banco de dados', and 'Fechar banco de dados'. Below the toolbar are tabs for 'Estrutura do banco de dados', 'Navegar dados', 'Editar pragmas', and 'Executar SQL'. A second toolbar contains icons for file operations and execution. The bottom toolbar shows open query files: 'Query1', 'Query2_Count', 'Where_clause', 'Queries.sql', 'INNER_aliases', and 'Alter_drop'. The main text area displays an SQL query with line numbers 1 through 11 on the left margin.

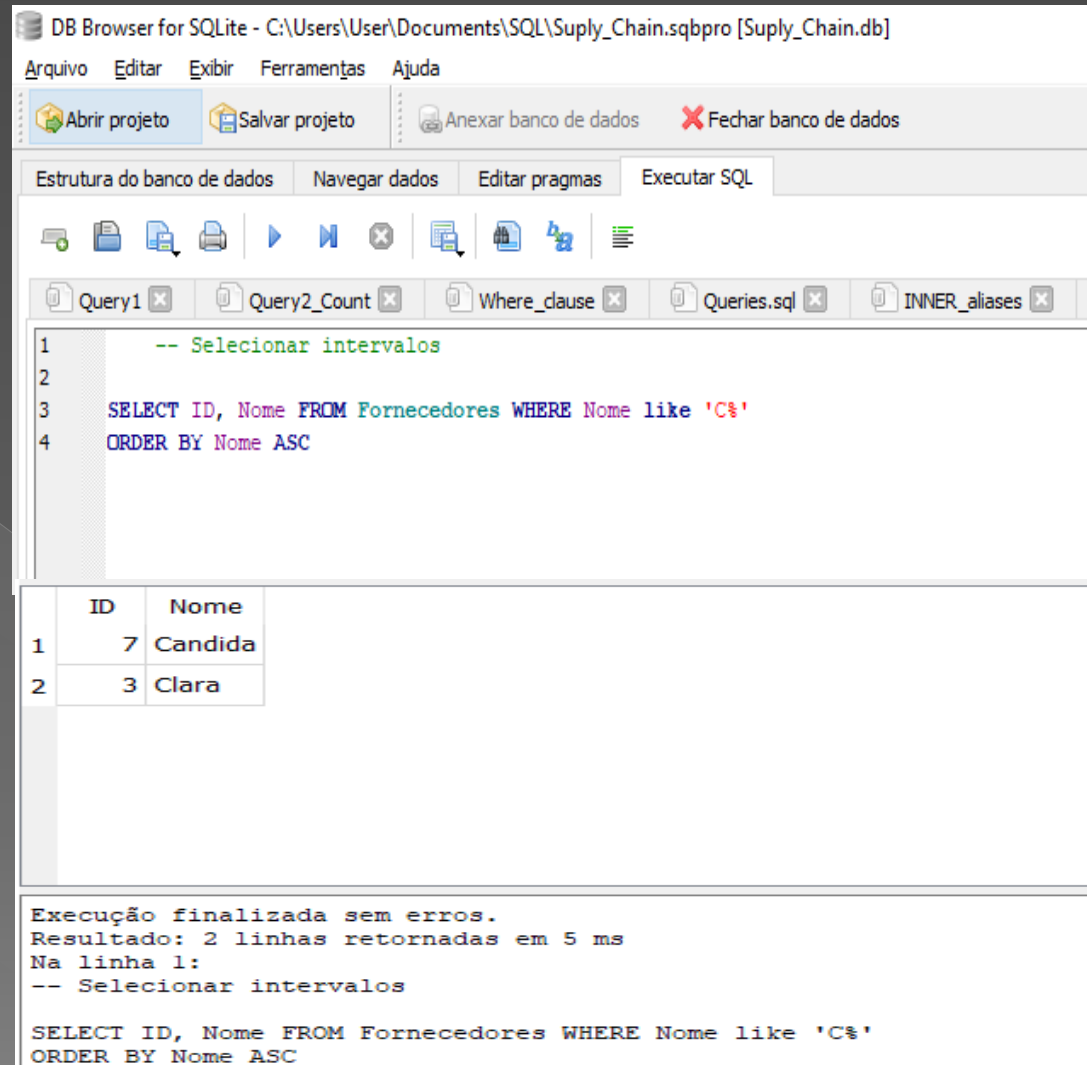
```
1  --Inserir múltiplas linhas
2
3  INSERT INTO Estoques
4      (Num_item, Pedido, Local, CustoUnidade, QTDE, Valor_Total, Nome_Equipamento, Responsavel)
5  VALUES
6      (006, 984, 'São Paulo', 930, 10, 9300, 'peças', 'Viviane'),
7      (007, 985, 'Curitiba', 831, 10, 8310, 'moedores', 'jose')
8
9
10
11
```

SELECIONAR INTERVALOS

ID e Nome, são colunas da tabela fornecedores.

Where funciona como um filtro e o caractere % é tido como “coringa” pois é equivalente a dizer “não me importo com os demais caracteres a seguir”, ele funciona com o operador like.

Order by ASC, irá classificar em ordem crescente as informações.



The screenshot shows the DB Browser for SQLite interface. The title bar indicates the database file is 'C:\Users\User\Documents\SQL\Suply_Chain.sqbp [Suply_Chain.db]'. The menu bar includes 'Arquivo', 'Editar', 'Exibir', 'Ferramentas', and 'Ajuda'. The toolbar has buttons for 'Abrir projeto', 'Salvar projeto', 'Anexar banco de dados', and 'Fechar banco de dados'. The main window has tabs for 'Estrutura do banco de dados', 'Navegar dados', 'Editar pragmas', and 'Executar SQL'. The 'Executar SQL' tab is active, showing a query editor with the following SQL code:

```
1      -- Selecionar intervalos
2
3      SELECT ID, Nome FROM Fornecedores WHERE Nome like 'C%'
4      ORDER BY Nome ASC
```

Below the query editor, the results are displayed in a table with two columns: 'ID' and 'Nome'. The table contains two rows of data:

	ID	Nome
1	7	Candida
2	3	Clara

At the bottom of the interface, a status bar shows the execution results: 'Execução finalizada sem erros. Resultado: 2 linhas retornadas em 5 ms'. Below this, the executed SQL query is repeated: 'Na linha 1: -- Selecionar intervalos' followed by 'SELECT ID, Nome FROM Fornecedores WHERE Nome like 'C%' ORDER BY Nome ASC'.

SELECIONAR DISTINTO

DB Browser for SQLite - C:\Users\User\Documents\SQL\Suply_Chain.sqbpro [Suply_Chain.db]

Arquivo Editar Exibir Ferramentas Ajuda

Abrir projeto Salvar projeto Anexar banco de dados Fechar banco de dados

Estrutura do banco de dados Navegar dados Editar pragmas Executar SQL

Query1 Query2_Count Where_clause Queries.sql INNER_aliases

```
1  --Grouping results sets
2
3  SELECT DISTINCT(Responsavel) FROM Estoques
```

	Responsavel
1	Karen
2	João
3	Jose
4	Viviane

Execução finalizada sem erros.
Resultado: 4 linhas retornadas em 5 ms
Na linha 1:
--Grouping results sets

SELECT DISTINCT(Responsavel) FROM Estoques

NESTE CASO OS RESPONSÁVEIS CONSTAM DIVERSAS VEZES NA TABELA. "DISTINCT" SERVE PARA APRESENTAR O RESPONSÁVEL APENAS UMA ÚNICA VEZ.

FUNÇÕES CASE, WHEN E ELSE

DB Browser for SQLite - C:\Users\User\Documents\SQL\Supply_Chain.sqbpro [Supply_Chain.db]

Arquivo Editar Exibir Ferramentas Ajuda

Abrir projeto Salvar projeto Anexar banco de dados Fechar banco de dados

Estrutura do banco de dados Navegar dados Editar pragmas Executar SQL

Queries.sql INNER_aliases Alter_drop Alter_ADD Update Insert

```
1  --case
2
3  SELECT Nome, Email,
4  CASE CustoMensalOperação
5  WHEN CustoMensalOperação > 1000 THEN 'Custo alto apresentar detalhamento e notas fiscais'
6  WHEN CustoMensalOperação < 700 THEN 'Apresentar relatório Mensal'
7  ELSE
8  'arquivar'
9  END AS Classe_Custo
10 FROM Fornecedores
11 ORDER BY Nome;
```

	Nome	Email	Classe_Custo
1	Candida	Soares	arquivar
2	Clara	nnevcç@gmail.com	arquivar
3	Danilo	Pereira	arquivar
4	Joana	Freire@k.mail.com	arquivar
5	Joao	jatun@yomail.com	arquivar
6	Jose	Fontes	arquivar

Execução finalizada sem erros.
Resultado: 9 linhas retornadas em 6 ms
Na linha 1:
--case
SELECT Nome, Email,

A ESTRUTURA DESTES COMANDOS É DEFINIDA POR CASE, SEGUIDA DE WHEN E ELSE. SIMILAR A IF E ELSE EM OUTRAS LINGUAGENS, TALS COMO PYTHON.

UNION ALL : SOMA DADOS DE DIFERENTES TABELAS

DB Browser for SQLite - C:\Users\User\Documents\SQL\Suply_Chain.sqbpro [Suply_Chain.sqbpro]

Arquivo Editar Exibir Ferramentas Ajuda

Abrir projeto Salvar projeto Anexar banco de dados Fechar banco de dados

Estrutura do banco de dados Navegar dados Editar pragmas Executar SQL

prop Alter_ADD Update Insert Select_intervalos

```
1 --Sum -somando duas tabelas
2
3 SELECT sum(Salario)AS Total FROM Funcionários
4 UNION ALL
5 SELECT sum(Pagamento_Mensal)AS total FROM Fornecedores
6
```

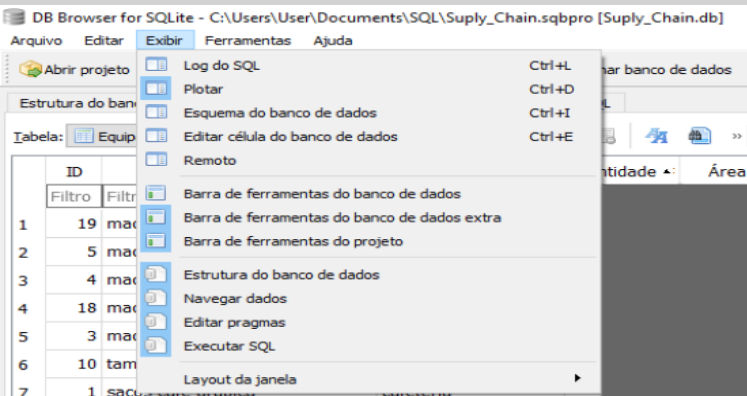
	Total
1	33369.3
2	71800

Execução finalizada sem erros.
Resultado: 2 linhas retornadas em 29 ms
Na linha 1:
--Sum -somando duas tabelas

```
SELECT sum(Salario)AS Total FROM Funcionários
UNION ALL
SELECT sum(Pagamento_Mensal)AS total FROM Fornecedores
```

PLOTAGEM

É possível fazer plotagem com uma tabela, através do comando **exibir, plotar**, conforme imagem à esquerda :



DB Browser for SQLite - C:\Users\User\Documents\SQL\Suply_Chain.sqbp [Suply_Chain.db]

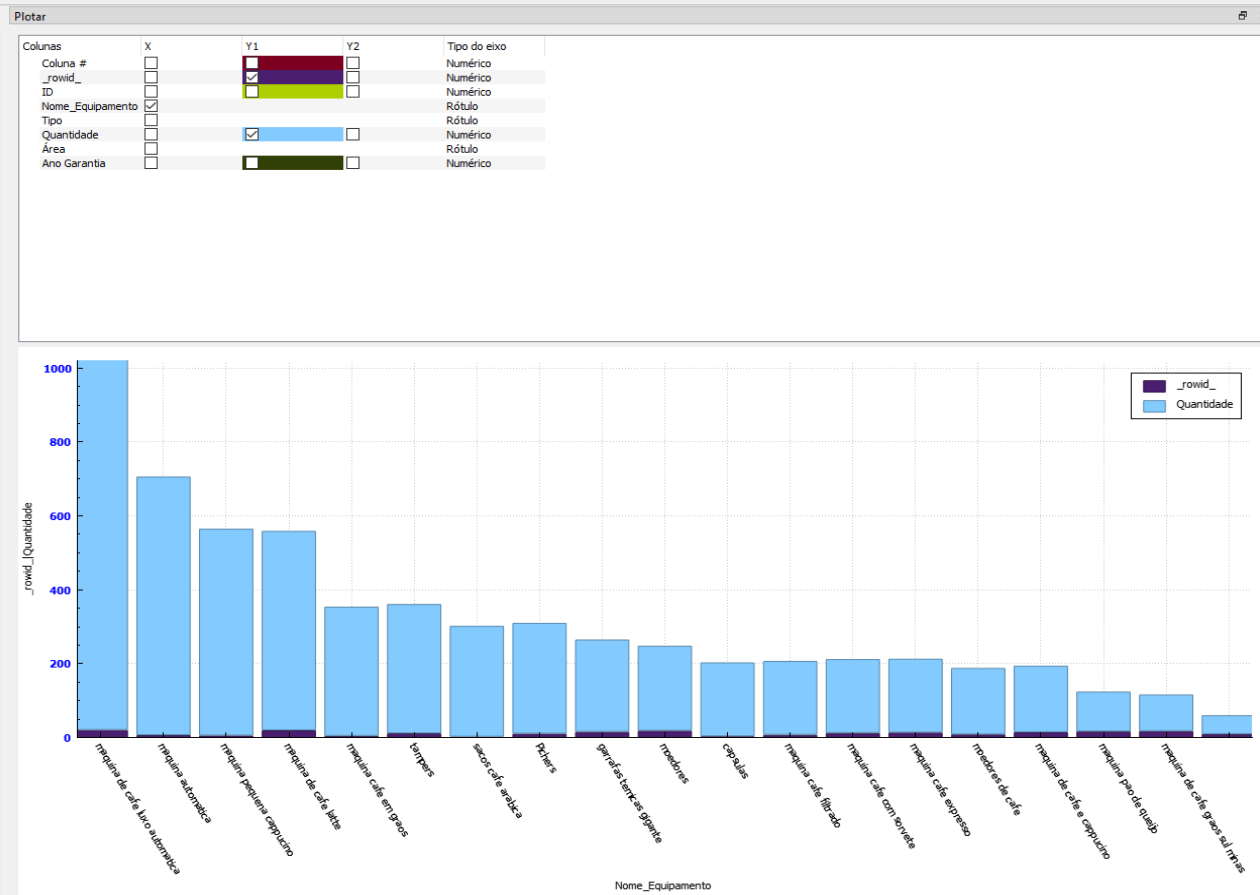
Arquivo Editar Exibir Ferramentas Ajuda

Abzir projeto

Estrutura do banco de dados

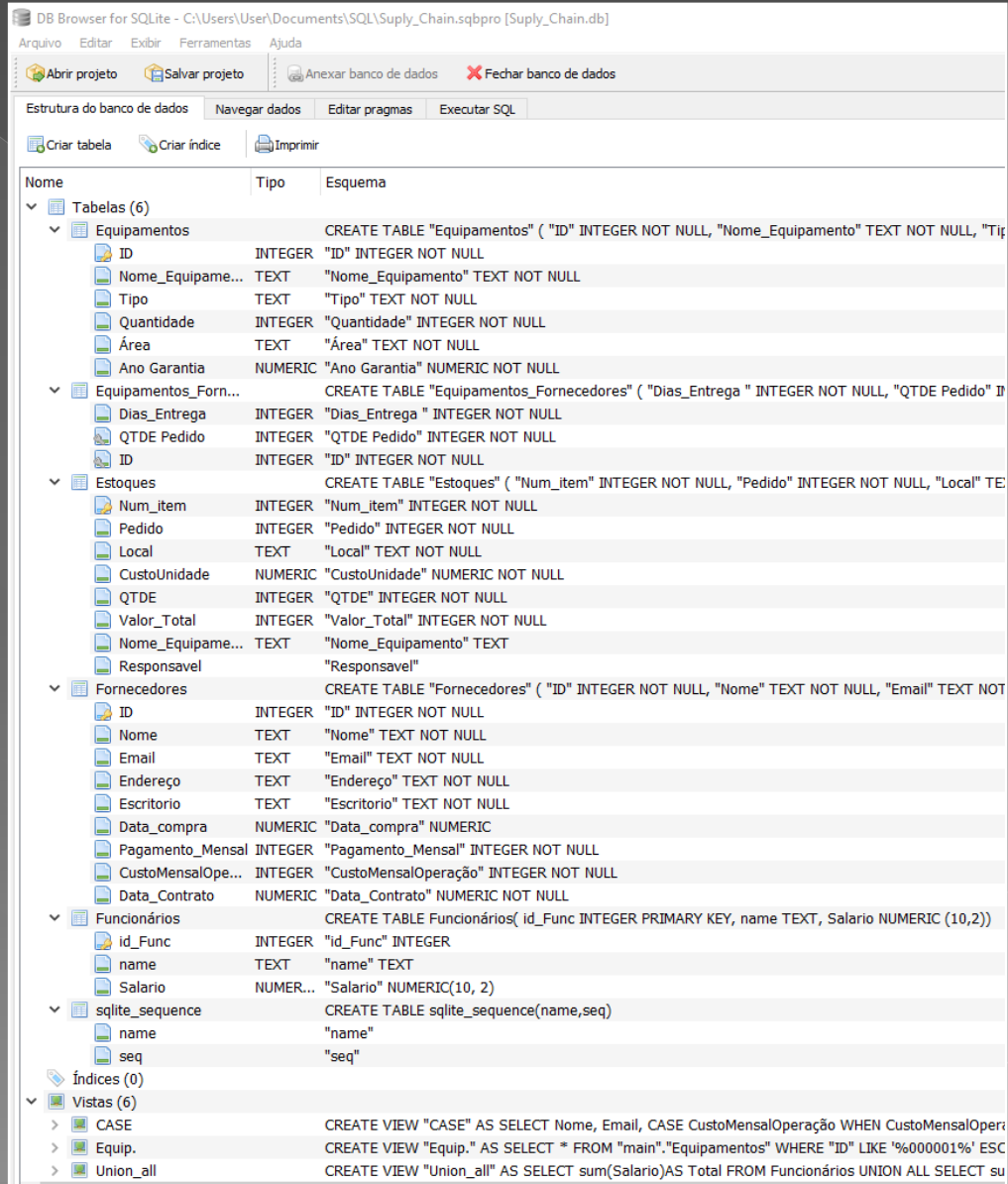
Tabela: Equip

ID	Nome_Equipamento	Tipo	Quantidade	Área	Ano Garantia
1	19 maquina de cafe luxu automatica	industrial	1020	empresas	6
2	5 maquina automatica	predios empresariais	700	C	6
3	4 maquina pequena cappuccino	residencia	560	B	3
4	18 maquina de cafe latte	gourmet	540	residencial	5
5	3 maquina cafe em graos	cafeteria	350	A	5
6	10 tampers	cafeteria	350	A	4
7	1 sacos cafe arabica	cafeteria	300	D	1
8	9 Pichers	cafeteria	300	A	5
9	14 garrafas termicas gigante	industrial	250	empresa	1
10	17 moedores	Industrial	230	empresas	8
11	2 capsulas	cafeteria	200	A	1
12	6 maquina cafe filtrado	predio empresarial	200	C	5
13	11 maquina cafe com sorvete	cafeteria	200	Vip	6
14	12 maquina cafe expresso	grãos	200	residencia	5
15	7 moedores de cafe	cafeteria	180	A	3
16	13 maquina de cafe e cappuccino	Gourmet	180	residencia	4
17	15 maquina pao de queijo	industrial	108	cafeteria	5
18	16 maquina de cafe graos sul minas	cafeteria	99	cafeteria	5
19	8 garrafas termicas	residencia	51	D	1



CONCLUSÃO

Neste projeto, Suply_Chain.db, inseri dados fictícios simulando uma empresa fabricante de equipamentos e máquinas de café. Desenvolvi 6 tabelas e pratiquei diversos comandos e funções.



Nome	Tipo	Esquema
▼ Tabelas (6)		
▼ Equipamentos		CREATE TABLE "Equipamentos" ("ID" INTEGER NOT NULL, "Nome_Equipamento" TEXT NOT NULL, "Tipo
ID	INTEGER	"ID" INTEGER NOT NULL
Nome_Equipame...	TEXT	"Nome_Equipamento" TEXT NOT NULL
Tipo	TEXT	"Tipo" TEXT NOT NULL
Quantidade	INTEGER	"Quantidade" INTEGER NOT NULL
Área	TEXT	"Área" TEXT NOT NULL
Ano Garantia	NUMERIC	"Ano Garantia" NUMERIC NOT NULL
▼ Equipamentos_Forn...		CREATE TABLE "Equipamentos_Fornecedores" ("Dias_Entrega " INTEGER NOT NULL, "QTDE Pedido" IN
Dias_Entrega	INTEGER	"Dias_Entrega " INTEGER NOT NULL
QTDE Pedido	INTEGER	"QTDE Pedido" INTEGER NOT NULL
ID	INTEGER	"ID" INTEGER NOT NULL
▼ Estoques		CREATE TABLE "Estoques" ("Num_item" INTEGER NOT NULL, "Pedido" INTEGER NOT NULL, "Local" TE
Num_item	INTEGER	"Num_item" INTEGER NOT NULL
Pedido	INTEGER	"Pedido" INTEGER NOT NULL
Local	TEXT	"Local" TEXT NOT NULL
CustoUnidade	NUMERIC	"CustoUnidade" NUMERIC NOT NULL
QTDE	INTEGER	"QTDE" INTEGER NOT NULL
Valor_Total	INTEGER	"Valor_Total" INTEGER NOT NULL
Nome_Equipame...	TEXT	"Nome_Equipamento" TEXT
Responsavel		"Responsavel"
▼ Fornecedores		CREATE TABLE "Fornecedores" ("ID" INTEGER NOT NULL, "Nome" TEXT NOT NULL, "Email" TEXT NOT
ID	INTEGER	"ID" INTEGER NOT NULL
Nome	TEXT	"Nome" TEXT NOT NULL
Email	TEXT	"Email" TEXT NOT NULL
Endereço	TEXT	"Endereço" TEXT NOT NULL
Escritorio	TEXT	"Escritorio" TEXT NOT NULL
Data_compra	NUMERIC	"Data_compra" NUMERIC
Pagamento_Mensal	INTEGER	"Pagamento_Mensal" INTEGER NOT NULL
CustoMensalOpe...	INTEGER	"CustoMensalOperação" INTEGER NOT NULL
Data_Contrato	NUMERIC	"Data_Contrato" NUMERIC NOT NULL
▼ Funcionários		CREATE TABLE Funcionários(id_Func INTEGER PRIMARY KEY, name TEXT, Salario NUMERIC (10,2))
id_Func	INTEGER	"id_Func" INTEGER
name	TEXT	"name" TEXT
Salario	NUMER...	"Salario" NUMERIC(10, 2)
▼ sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
name		"name"
seq		"seq"
Índices (0)		
▼ Vistas (6)		
> CASE		CREATE VIEW "CASE" AS SELECT Nome, Email, CASE CustoMensalOperação WHEN CustoMensalOper
> Equip.		CREATE VIEW "Equip." AS SELECT * FROM "main"."Equipamentos" WHERE "ID" LIKE '%000001%' ESC
> Union_all		CREATE VIEW "Union_all" AS SELECT sum(Salario)AS Total FROM Funcionários UNION ALL SELECT su