



SQL

Linguagem de Definição de Dados

Prof. Humberto Razente

Bloco B - sala 1B144

SQL

Structured Query Language

- ◆ Desenvolvida e implementada pelo laboratório de pesquisa da IBM em San Jose – início da década de 70
- ◆ Inicialmente chamada de SEQUEL (*Structured English QUERy Language*)
- ◆ Criada como interface entre usuários e o primeiro SGBDR – SYSTEM R

SQL


Structured Query Language

- ◆ Uma das mais importantes linguagens relacionais
- ◆ Exemplos de SGBD relacionais que utilizam SQL
 - Oracle
 - Informix
 - Ingres
 - MS SQL Server
 - Interbase/Firebird
 - Sybase
 - DB2
 - MySQL/MariaDB
 - PostgreSQL

SQLite Home Page

Private browsing

https://www.sqlite.org/index.html



Small. Fast. Reliable.
Choose any three.

HomeAboutDocumentationDownloadLicenseSupportPurchaseSearch

What Is SQLite?

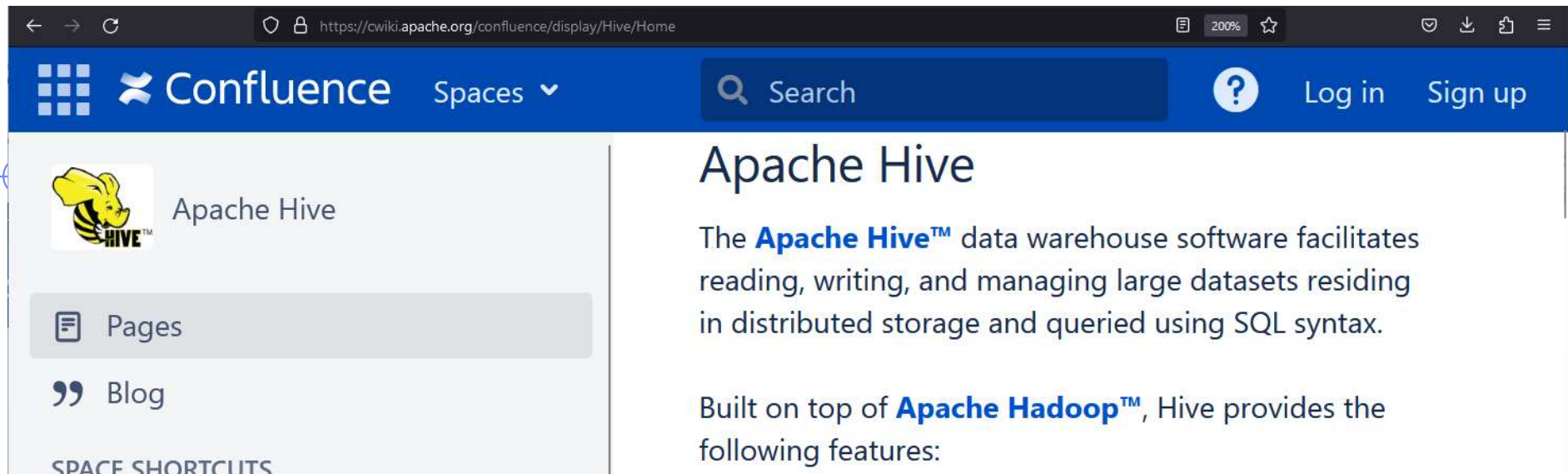
SQLite is a C-language library that implements a [small](#), [fast](#), [self-contained](#), [high-reliability](#), [full-featured](#), SQL database engine. SQLite is the [most used](#) database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day. [More Information...](#)

The SQLite [file format](#) is stable, cross-platform, and backwards compatible and the developers pledge to keep it that way [through the year 2050](#). SQLite database files are commonly used as containers to transfer rich content between systems [\[1\]](#) [\[2\]](#) [\[3\]](#) and as a long-term archival format for data [\[4\]](#). There are over 1 trillion (1e12) SQLite databases in active use [\[5\]](#).

SQLite [source code](#) is in the [public-domain](#) and is free to everyone to use for any purpose.

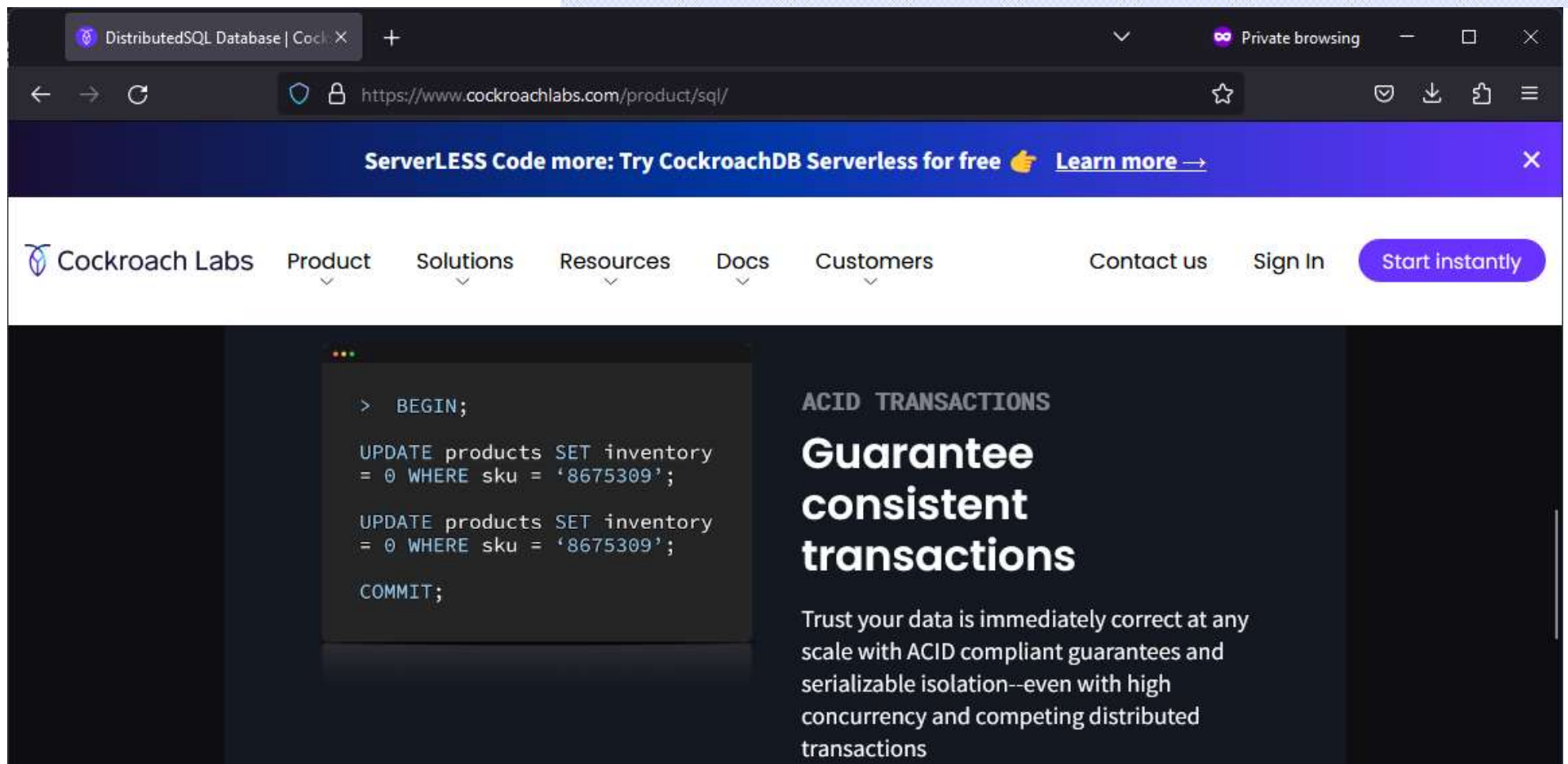
Common Links

- Features
- When to use SQLite
- Getting Started
- Try it live!
- Prior Releases
- SQL Syntax
 - Pragas
 - SQL functions
 - Date & time functions
 - Aggregate functions
 - Window functions
 - Math functions
 - JSON functions
- C/C++ Interface Spec
 - Introduction



◆ SQL em ambiente Map/Reduce

- Apache Hive → foco em consultas analíticas (OLAP)
 - <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL>
 - <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Select>



◆ NewSQL

■ CockroachDB

- ◆ Suporte a transações (OLTP distribuído)
- ◆ <https://www.cockroachlabs.com/product/sql/>

SQL

Structured Query Language

- ◆ Atrativo: pequena quantidade de comandos para realizar todas as operações necessárias para definição e manipulação de relações
 - Simplicidade
 - Grande poder de consulta

- ◆ Padrão facilita migração

SQL

Structured Query Language

◆ O padrão SQL

- *American National Standard Institute (ANSI) e International Organization for Standardization (ISO)*
- Versões:
 - ◆ SQL:2023 (ISO/IEC 9075:2023)
 - ◆ SQL 2016 (ISO/IEC 9075:2016)
 - ◆ SQL 2011
 - ◆ SQL 3 → SQL 99
 - ◆ SQL 2 → SQL 92
 - ◆ SQL 1 → SQL 86

Composição do SQL

◆ Linguagem de Definição dos Dados (DDL)

- comandos para a definição, a modificação e a remoção de relações, além da criação e da remoção de índices

◆ Linguagem Interativa de Manipulação dos Dados (DML)

- comandos para a consulta, a inserção, a remoção e a modificação de tuplas no banco de dados

Composição do SQL

◆ Linguagem de Manipulação dos Dados Embutida

- pode ser utilizada a partir de linguagens de programação de propósito geral

◆ Definição de visões

- SQL DDL inclui comandos para a criação e a remoção de visões

◆ Restrições de integridade

- SQL DDL possui comandos para a especificação de restrições de integridade

Composição do SQL

◆ Autorização

- SQL DDL inclui comandos para a especificação de direitos de acesso a relações e visões

◆ Gerenciamento de transações

- introduz comandos para a especificação do início e do fim das transações

◆ Recuperação de falhas

- introduz comandos para utilização do arquivo de *log*

Composição do SQL

◆ Linguagem de Definição dos Dados (DDL)

- CREATE
- ALTER
- DROP

SQL DDL

◆ CREATE DATABASE | SCHEMA

- cria um esquema de BD relacional

◆ DROP DATABASE | SCHEMA

- remove um esquema de BD relacional

CREATE DATABASE

```
CREATE {DATABASE | SCHEMA} nome  
    [USER `username` [PASSWORD `password` ]  
    ... ;
```

- ◆ Cria um esquema de BD relacional
 - agrupa as tabelas e outros comandos que pertencem à mesma aplicação
 - identifica o proprietário do esquema
- ◆ Característica
 - o esquema inicial não possui tabelas/dados

DROP DATABASE

```
DROP DATABASE {DATABASE | SCHEMA} nome  
[CASCADE | RESTRICT] ;
```

◆ Remove um esquema de BD relacional

- tabelas/dados
 - índices
 - arquivos de log
- } todos os elementos associados

◆ Usuários autorizados

- proprietário do banco de dados
- DBA ou usuário com privilégio de *root*

DROP DATABASE

◆ CASCADE

- remove um esquema de BD, incluindo todas as suas tabelas e os seus outros elementos

◆ RESTRICT

- remove um esquema de BD somente se não existirem elementos definidos para esse esquema

Curiosidade

◆ A recuperação de metadados é feita pela execução de funções ou de consultas a visões ou variáveis globais

- definidas por cada fabricante



SQL DDL

◆ CREATE TABLE

- cria uma nova tabela (relação) no BD
- a nova tabela não possui dados

◆ DROP TABLE

- remove uma tabela (relação) e todas as suas instâncias do BD

◆ ALTER TABLE

- altera a estrutura de uma tabela (relação) já existente no BD

CREATE TABLE

```
CREATE TABLE nome_tabela ( A1 D1 R1,  
                             A2 D2 R2,  
                             ...  
                             An Dn Rn );
```

- ◆ Cria uma nova tabela (relação)
- ◆ Cria os atributos da nova tabela, com
 - nome do atributo: A_i ($1 \leq i \leq n$)
 - tipo de dado (domínio do atributo): D_i
 - restrições que atuam no atributo: R_i

Exemplos de Tipos de Dados

◆ Numéricos

- smallint | integer | float | double precision
- decimal | numeric

◆ Hora/Data

- date | time | timestamp

◆ Boolean

- TRUE/FALSE ou UNKNOWN (NULL)

◆ Cadeias de Caracteres

- char | character | varchar | ...

◆ Outros

- blob

SGBDs apresentam particularidades. Exemplo:
Oracle: boolean presente apenas em PL/SQL
Postgres: boolean presente no SQL
• CREATE TABLE x (y boolean)

Exemplos de Tipos de Dados

- ◆ <https://www.postgresql.org/docs/current/datatype.html>
- ◆ <https://docs.oracle.com/en/database/oracle/oracle-database/23/development.html>
 - SQL Language Reference
 - ◆ <https://docs.oracle.com/en/database/oracle/oracle-database/23/sqlrf/Data-Types.html>
- ◆ <https://www.ibm.com/docs/en/db2-for-zos/13?topic=tables-db2-table-columns>
- ◆ <https://learn.microsoft.com/pt-br/sql/sql-server/?view=sql-server-ver16>
- ◆ <https://sqlite.org/about.html>

Restrições de Integridade

Atributo

◆ Valor nulo

- representado por NULL
- membro de todos os domínios

◆ Restrição NOT NULL

- especificada quando NULL não é permitido
- proíbe que o atributo receba valor nulo

◆ Comparações em consultas

- usar IS NULL e IS NOT NULL

Restrições de Integridade

Atributo

◆ Cláusula DEFAULT

- associa um valor *default* para um atributo, caso nenhum outro valor seja especificado
 - ◆ UF char(2) default 'MG'

◆ Cláusula CHECK

- especifica um predicado que precisa ser satisfeito por todas as tuplas de uma relação
- exemplos
 - ◆ saldo int CHECK (saldo \geq 0)
 - ◆ nível char(15) CHECK (nível IN ('Bacharelado', 'Mestrado', 'Doutorado'))

Restrições de Integridade

Chave

◆ Cláusula PRIMARY KEY

- identifica os atributos da relação que formam a sua chave primária
 - ◆ os atributos são implicitamente NOT NULL
- sintaxe

PRIMARY KEY (atributo₁, atributo₂, ..., atributo_x)

◆ Cláusula UNIQUE

- não permite valores duplicados para um determinado atributo

Restrições de Integridade

Chave

◆ Integridade referencial

- dependência existente entre a chave estrangeira de uma relação e a chave primária da relação relacionada
- trata dos seguintes problemas
 - ◆ atualização ou exclusão de elementos da chave primária sem fazer um ajuste coordenado nas chaves estrangeiras
 - ◆ inclusão ou alteração de valores não nulos na chave estrangeira que não existam na chave primária

Restrições de Integridade

Chave

◆ Cláusula FOREIGN KEY

■ características

- ◆ elimina a possibilidade de violação da integridade referencial
- ◆ reflete nas chaves estrangeiras todas as alterações na chave primária

■ sintaxe

FOREIGN KEY (atributos)

REFERENCES nome_relação (atributos)

[ON UPDATE [CASCADE | SET NULL | SET DEFAULT]]

[ON DELETE [CASCADE | SET NULL | SET DEFAULT]]

DROP TABLE

```
DROP TABLE nome_tabela ;
```

◆ Remove uma tabela existente do BD

- dados – metadados
- índices
- gatilhos que referenciam a tabela

◆ Usuários autorizados

- proprietário do banco de dados
- DBA ou usuário com privilégio de *root*

ALTER TABLE

```
ALTER TABLE nome_tabela;
```

◆ Altera o esquema de uma tabela do BD

- adiciona
- remove
- altera

} colunas ou restrições
de integridade

Exemplos: ALTER TABLE

```
ALTER TABLE nome_tabela  
  ADD (A1 D1 R1),  
  ...  
  ADD (An Dn Rn)
```

- inclui novas colunas na tabela

```
ALTER TABLE nome_tabela DROP A1
```

- elimina uma coluna já existente da tabela

Exemplos: ALTER TABLE

ALTER TABLE nome_tabela ALTER [COLUMN] A₁ TO A₂

- modifica o nome de uma coluna existente de A₁ para A₂

ALTER TABLE nome_tabela
ALTER [COLUMN] A₁ TYPE NOVOTIPO

- modifica o tipo de dado de uma coluna

SQL DDL

◆ CREATE DOMAIN

- cria um domínio para um tipo de dados

◆ DROP DOMAIN

- remove um domínio existente do BD

◆ ALTER DOMAIN

- altera a definição de domínio

CREATE DOMAIN

```
CREATE DOMAIN nome_domínio [AS] tipo_dado  
    [DEFAULT ... ]  
    [NOT NULL]  
    [CHECK ...]  
    ... ;
```

- ◆ Cria um domínio para um tipo de dados
 - restrições de integridade
- ◆ Característica
 - a definição do domínio é global ao BD

Particularidades:
create domain é a sintaxe usada no PostgreSQL.
create type é a sintaxe usada no Oracle.

DROP DOMAIN

```
DROP DOMAIN nome_domínio ;
```

- ◆ Remove um domínio existente do BD
 - falha caso o domínio esteja definindo o tipo de dado de alguma coluna

ALTER DOMAIN

```
ALTER DOMAIN nome_domínio  
...;
```

- ◆ Altera um domínio existente do BD
 - remove ou define restrições de integridade

SQL DDL

◆ CREATE INDEX

- cria um índice sobre uma ou mais colunas de uma tabela

◆ DROP INDEX

- remove um índice existente do BD

◆ ALTER INDEX

- torna um índice ativo ou inativo

CREATE INDEX

```
CREATE [UNIQUE] INDEX nome_índice  
ON nome_tabela (nome_coluna [, ...] ) ;
```

- ◆ Cria um índice sobre uma ou mais colunas de uma tabela
- ◆ Considerações
 - desempenho das consultas *versus* custos de atualização e de armazenamento
- ◆ Chaves candidatas podem ser garantidas por índices do tipo UNIQUE

Índice

- ◆ Estrutura de acesso auxiliar usada para melhorar o desempenho na recuperação (pesquisa) de registros
- ◆ Pesquisa
 - restringida a um subconjunto dos registros, em contrapartida à análise do conjunto completo

Índice

◆ Sobrecarga adicional

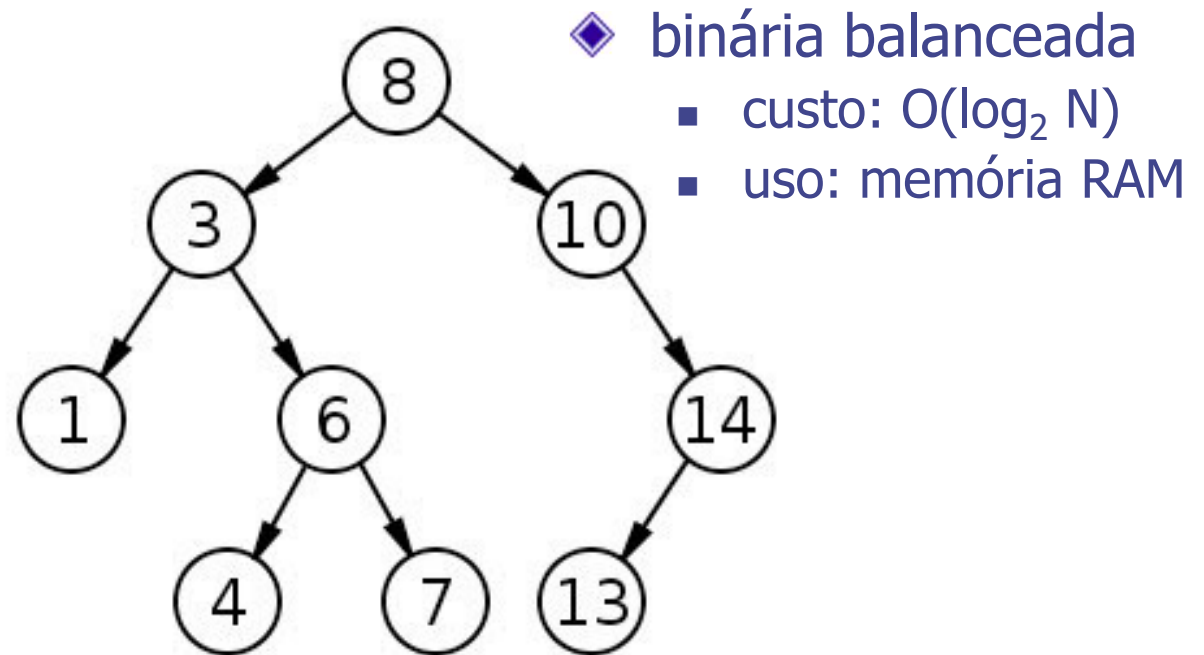
- atualização do índice ao se executar operações de insert, update e delete
- porém permite realizar consultas de modo otimizado

◆ Observações

- existe uma variedade de índices, cada qual com uma estrutura de dados particular
- qualquer atributo pode ser usado para criar um índice
- vários índices podem ser definidos para uma mesma tabela

Índice – Exemplo baseado em árvore

A	B
8	maria
3	joão
10	cecília
1	luiz
6	júlia
14	marina
4	enzo
7	rúbens
13	carlos



◆ Armazenáveis em disco:

◆ árvore B+: $O(\log_M N)$

◆ operadores relacionais

◆ hash: $O(1)$

◆ apenas consultas por igualdade

Atributos Indexados

◆ Chaves

- primárias e estrangeiras

◆ Presentes em operações de seleção

- valores requeridos em condições

- ◆ igualdade
- ◆ faixa de valores (i.e., *range queries*)

◆ Participam de condições de junção

um atributo deve
ser indexado?

algumas consultas podem ser processadas apenas
varrendo-se o índice, sem recuperar qualquer dado

Índice sobre Vários Atributos

◆ Condição

- vários atributos de uma relação estão envolvidos juntamente em diversas consultas

quando deve
ser criado?

◆ Restrição

- ordem dos atributos dentro do índice deve corresponder às consultas

◆ Exemplo

- índice sobre (estilo_carro, cor)

DROP INDEX

```
DROP INDEX nome_índice ;
```

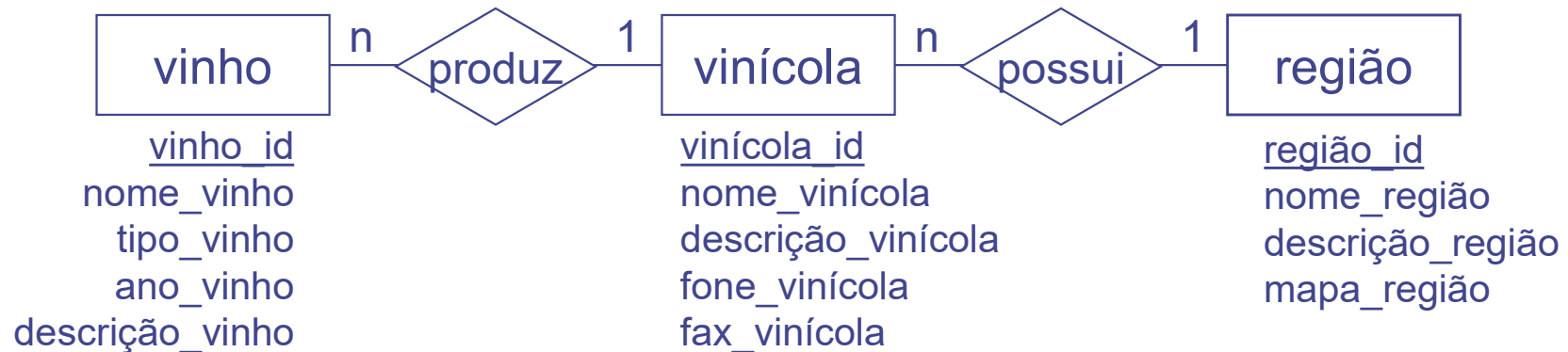
- ◆ Remove um índice existente do BD

ALTER INDEX

```
ALTER INDEX nome_índice;
```

- ◆ Altera a definição de um índice

Exemplo: loja de vinhos



- ◆ **região** (região_id, nome_região, mapa_região, descrição_região)
- ◆ **vinícola** (vinícola_id, nome_vinícola, descrição_vinícola, fone_vinícola, fax_vinícola, **região_id**)
- ◆ **vinho** (vinho_id, nome_vinho, tipo_vinho, ano_vinho, descrição_vinho, **vinícola_id**)

Exemplo

região (região_id, nome_região, mapa_região,
descrição_região)

```
CREATE TABLE região (  
    região_id          smallint      NOT NULL PRIMARY KEY ,  
    nome_região        varchar(100)  NOT NULL ,  
    mapa_região        blob          ,  
    descrição_região   blob          ,  
    PRIMARY KEY (região_id)  
);
```

restrição de coluna

restrição de tabela

Nota: chave primária deve ser especificada como restrição de colula ou de tabela (não se deve usar os dois modos na mesma tabela, pois significa que há duas chaves primárias definidas)

vinícola (vinícola_id, nome_vinícola, descrição_vinícola,
fone_vinícola, fax_vinícola, região_id)

```
CREATE TABLE vinícola (  
    vinícola_id      smallint      NOT NULL ,  
    nome_vinícola    varchar(100)  NOT NULL ,  
    descrição_vinícola blob          ,  
    fone_vinícola     varchar(15)   ,  
    fax_vinícola      varchar(15)   ,  
    região_id        smallint      NOT NULL ,  
    PRIMARY KEY (vinícola_id),  
    FOREIGN KEY (região_id) REFERENCES região(região_id)  
);
```

vinho (vinho_id, nome_vinho, tipo_vinho,
ano_vinho, descrição_vinho, vinícola_id)

```
CREATE TABLE vinho (  
    vinho_id          smallint      NOT NULL  
    nome_vinho        varchar(50)   NOT NULL  
    tipo_vinho        varchar(10)   DEFAULT 'seco ' NOT NULL,  
    ano_vinho         integer       CHECK (ano_vinho > 1970),  
    descrição_vinho   blob  
    vinícola_id       smallint      NOT NULL,  
    PRIMARY KEY (vinho_id)  
    UNIQUE (nome_vinho),  
    FOREIGN KEY (vinícola_id) REFERENCES vinícola(vinícola_id)  
);
```

SQL DML

Inserção

- ◆ Realizada através da especificação
 - de uma tupla particular
 - de uma consulta que resulta em um conjunto de tuplas a serem inseridas
- ◆ Valores dos atributos das tuplas inseridas
 - devem pertencer ao domínio do atributo
- ◆ Atributos sem valores
 - especificados por NULL ou valor DEFAULT

SQL DML

Inserção

```
INSERT INTO nome_tabela  
VALUES ( V1, V2, ..., VN );
```

- ◆ Ordem dos atributos deve ser mantida

SQL DML

Inserção

```
INSERT INTO nome_tabela (A1, A2, ..., An)  
VALUES ( V1, V2, ..., VN );
```

- Ordem dos atributos não precisa ser mantida

SQL DML

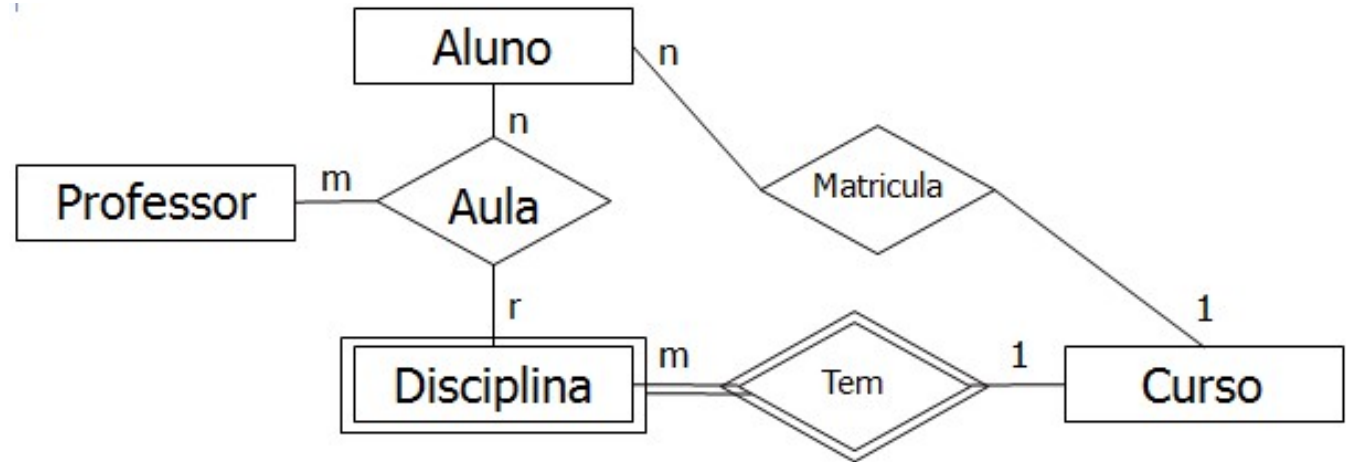
Inserção - Exemplos

- ◆ INSERT INTO região
VALUES (12, 'nome região', NULL,
 'descrição');
- ◆ INSERT INTO região (região_id,
 nome_região)
VALUES (12, 'nome região');

Exercício

Crie o esquema do modelo em SQL:

Desafio: fazer no



Modelo Relacional:

Curso = { NumCurso, Nome, TotalCréditos }

Disciplina = { NumCurso, NumDisp, Nome, QuantCreditos }

Aula = { NumAluno, NumCurso, NumDisp, NumFunc, Semestre, Nota }

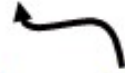
Aluno = { NumAluno, Nome, Endereco, Cidade, Telefone, NumCurso }

Professor = { NumFunc, Nome, Admissao, AreaPesquisa }

Curso = { NumCurso, Nome, TotalCréditos }

```
create table curso (  
    NumCurso integer primary key,  
    Nome varchar(50),  
    TotalCreditos integer  
);
```

Curso = { NumCurso, Nome, TotalCréditos }



Disciplina = { NumCurso, NumDisp, Nome, QuantCreditos }

```
create table disciplina (  
    NumCurso integer,  
    NumDisp integer,  
    Nome varchar(50),  
    QuantCreditos integer,  
    primary key (NumCurso, NumDisp),  
    foreign key (NumCurso) references curso (NumCurso)  
);
```

Curso = { NumCurso, Nome, TotalCréditos }

Disciplina = { NumCurso, NumDisp, Nome, QuantCreditos }

Aluno = { NumAluno, Nome, Endereco, Cidade, Telefone, NumCurso }



```
create table aluno (  
    NumAluno integer primary key,  
    Nome varchar(50),  
    Endereco varchar(50),  
    Cidade varchar(50),  
    Telefone char(15),  
    NumCurso integer,  
    foreign key (NumCurso) references Curso(NumCurso)  
);
```

Curso = { NumCurso, Nome, TotalCréditos }

Disciplina = { NumCurso, NumDisp, Nome, QuantCreditos }



Aluno = { NumAluno, Nome, Endereco, Cidade, Telefone, NumCurso }

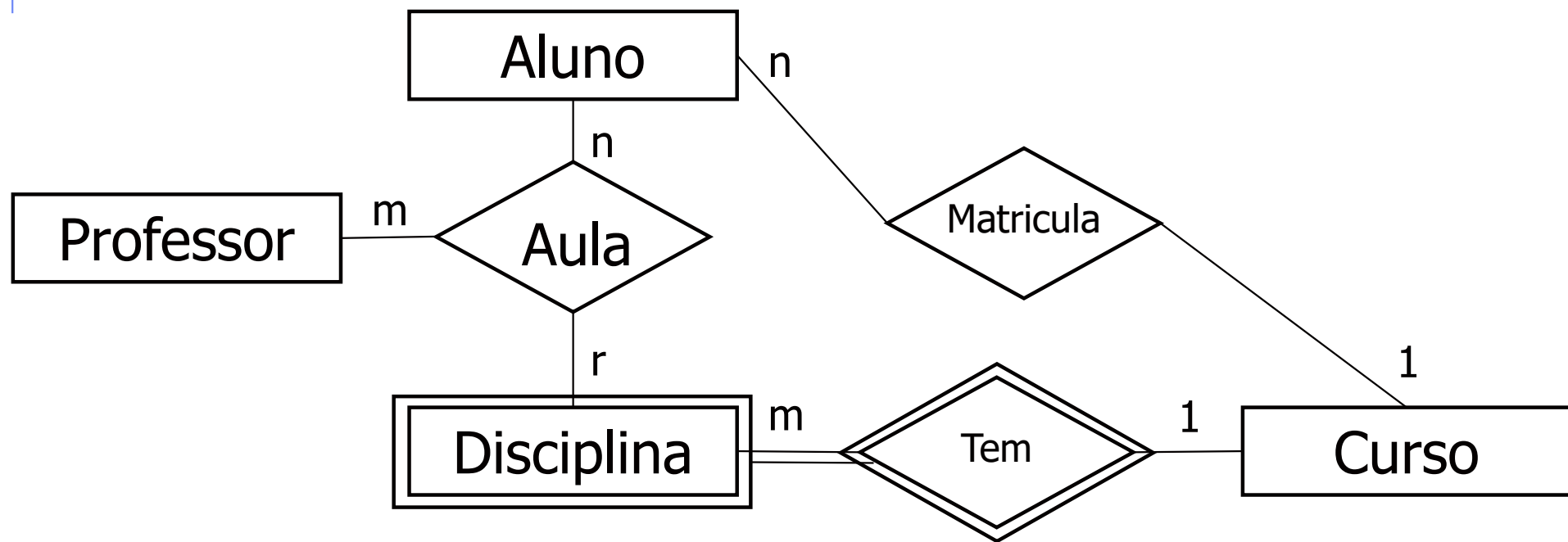
Professor = { NumFunc, Nome, Admissao, AreaPesquisa }

```
create table professor (  
    NumFunc integer primary key,  
    Nome varchar(50),  
    Admissao date,  
    AreaPesquisa varchar(20)  
);
```

Curso = { NumCurso, Nome, TotalCréditos }
 Disciplina = { NumCurso, NumDisp, Nome, QuantCreditos }
 Aula = { NumAluno, NumCurso, NumDisp, NumFunc, Semestre, Nota }
 Aluno = { NumAluno, Nome, Endereco, Cidade, Telefone, NumCurso }
 Professor = { NumFunc, Nome, Admissao, AreaPesquisa }

```

create table Aula (
  NumAluno integer, NumCurso integer,
  NumDisp integer, NumFunc integer,
  Semestre integer, Nota integer,
  primary key (NumAluno, NumCurso, NumDisp, NumFunc,
    Semestre),
  foreign key (NumAluno) references Aluno (NumAluno),
  foreign key (NumCurso, NumDisp) references Disciplina
    (NumCurso, NumDisp),
  foreign key (NumFunc) references Professor (NumFunc)
);
  
```



Bibliografia

- ◆ Elmasri, Ramez; Navathe, Shamkant B. **Sistemas de banco de dados.** 6ª ed. São Paulo: Addison Wesley, 2011
 - capítulo 4: SQL Básica