

Linguagem SQL - DDL

Base de Dados - 2023/24

Carlos Costa

Linguagem SQL

- Structured Query Language (SQL)
 - SEQUEL
- Linguagem para definir, manipular e questionar uma Base de Dados Relacional.
 - É uma linguagem orientada ao processamento de conjuntos
- 2 sublinguagens principais
 - DDL - Data Definition Language
 - DML - Data Manipulation Language
- 1 sublinguagem de controlo BD
 - DCL - Data Control Language

SQL - Versões

- 1986 (SQL-86 e SQL-87)
 - Publicado pela ANSI e ratificado pela ISO.
- 1989 (SQL-89)
- 1992 (SQL-92)
 - conhecido como SQL2.
- 1999 (SQL:1999)
 - conhecido como SQL 3.
 - inclui expressões regulares, queries recursivas, triggers, tipos não escalares, procedimentos, funcionalidades orientadas a objectos, etc.
- 2003 (SQL:2003)
 - Inclui suporte a XML e colunas com numeração automática.
- 2006 (SQL:2006)
 - Define formas de interacção SQL-XML: como importar e armazenar XML em BD SQL, XQuery, etc.
- 2008
- 2011

SQL - SQL Server

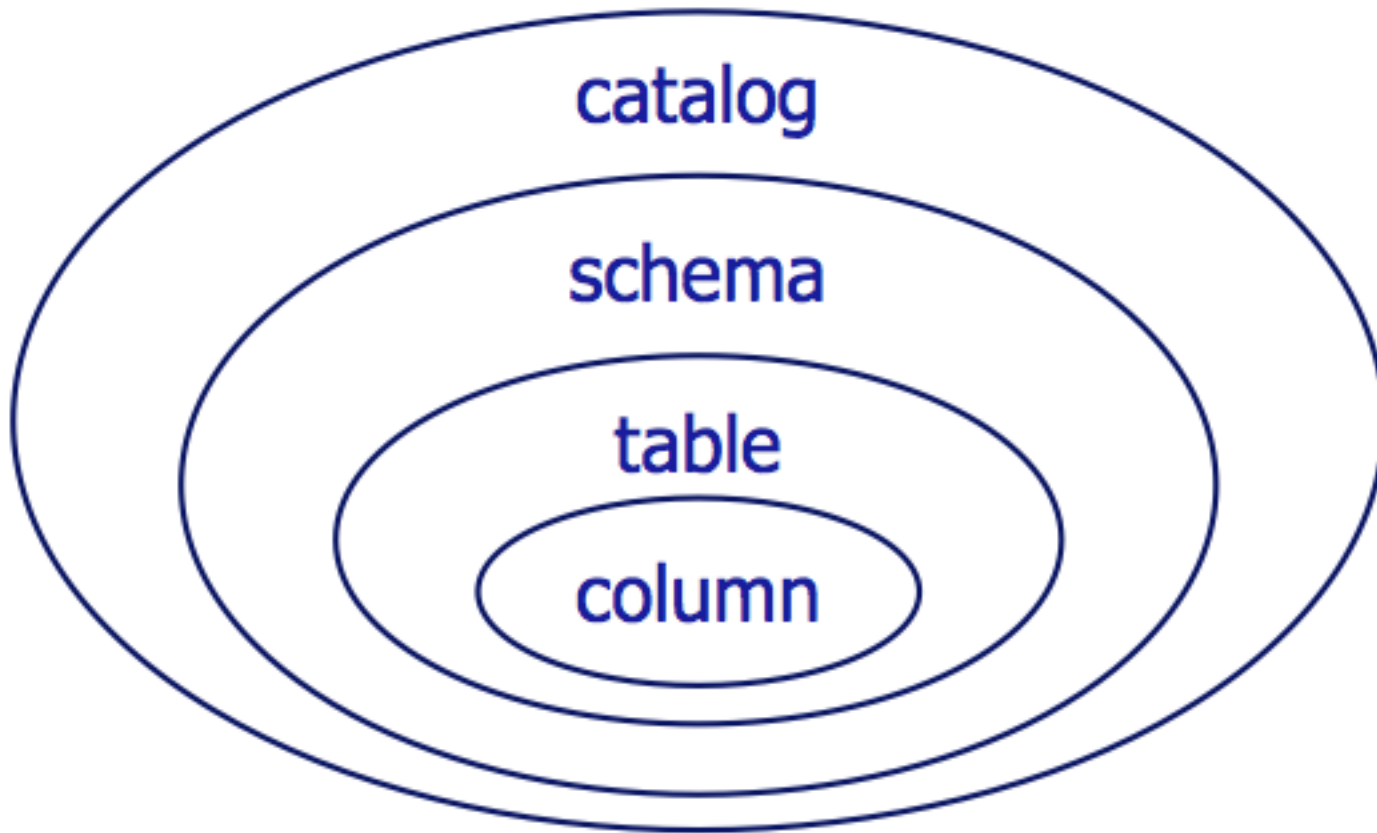
- Vamos utilizar, como ferramenta de trabalho, a versão SQL Server (≥ 2014)

Transact-SQL

“Microsoft SQL Server team has extended the ANSI definition with several enhancements and new commands, and has left out a few commands because SQL Server implemented them differently. The result is Transact-SQL, or T-SQL — the dialect of SQL understood by SQL Server”

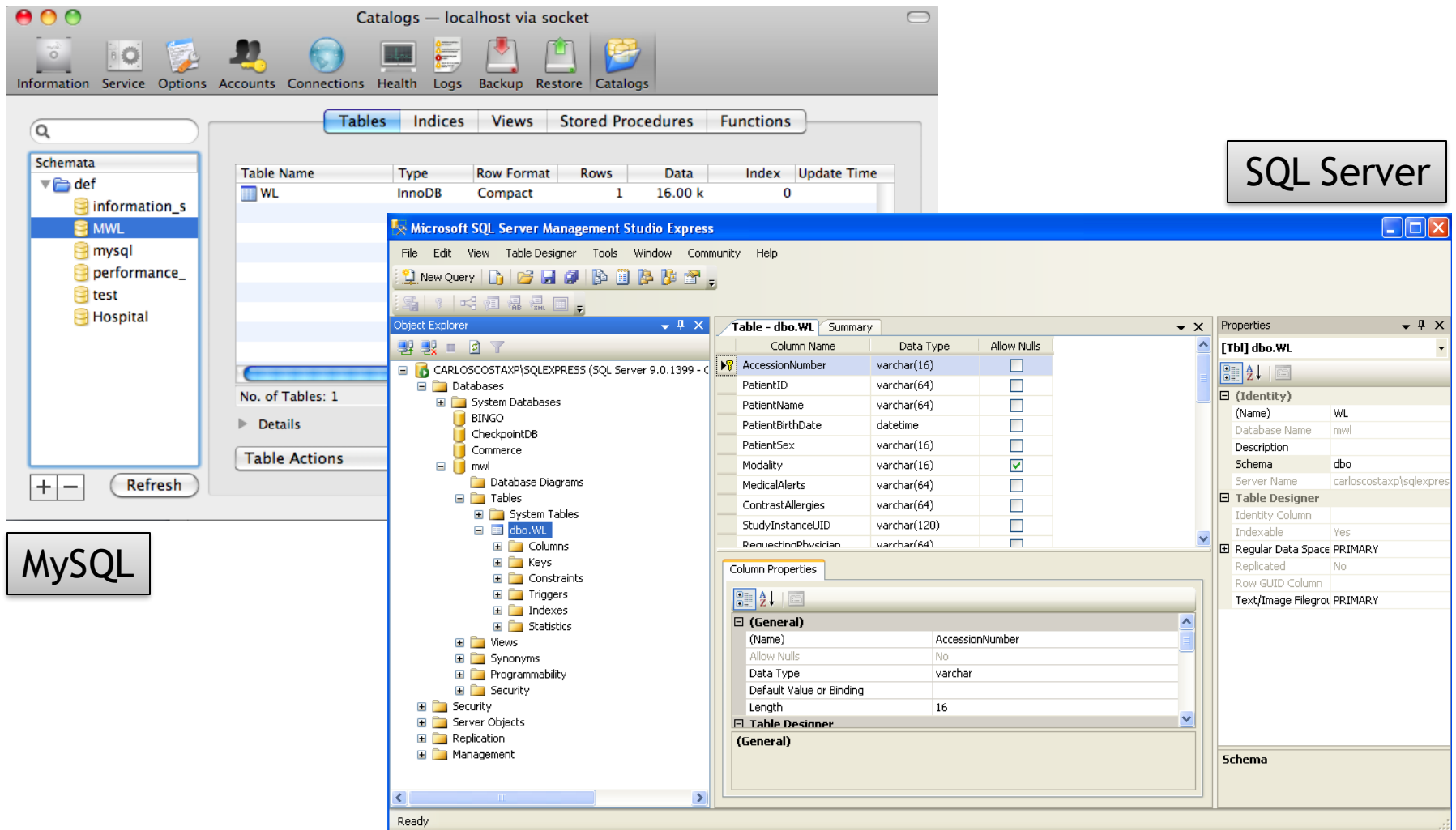
“Missing from T-SQL are very few ANSI SQL commands, primarily because Microsoft implemented the functionality in other ways.”

SQL - Hierarquia de Objetos



Mas há mais elementos como, por exemplo, triggers, vistas, índices, stored procedures, funções, etc.

SQL - catalog, schema e database



The image displays two database management interfaces side-by-side. On the left is the MySQL 'Catalogs' window, showing a tree view of schemas (def, information_s, MWL, mysql, performance_, test, Hospital) and a table list for the 'WL' table. On the right is Microsoft SQL Server Management Studio Express, showing the 'Object Explorer' with a tree view of databases (CARLOSCOSTAXP\SQLEXPRESS) and a table list for the 'dbo.WL' table. The 'Table Designer' for 'dbo.WL' is open, showing columns: AccessionNumber (varchar(16)), PatientID (varchar(64)), PatientName (varchar(64)), PatientBirthDate (datetime), PatientSex (varchar(16)), Modality (varchar(16)), MedicalAlerts (varchar(64)), ContrastAllergies (varchar(64)), StudyInstanceUID (varchar(120)), and ReferringPhysician (varchar(64)). The 'Properties' window for '[Tbl] dbo.WL' is also open, showing details like Identity Column, Indexable, Regular Data Space, Replicated, Row GUID Column, and Text/Image Filegroup.

MySQL

SQL Server

O significado destes termos varia de acordo com SGBD

SQL Server: database_name . schema_name . table_name

SQL - Notas introdutórias

- SQL utiliza...
tabela, linha e coluna (table, row and column)
... para designar os termos formais:
relação, tuplo e atributo do modelo relacional
- Cada instrução SQL termina com um ponto e vírgula (“;”)
- Comentar um linha “--”
- Comentar um bloco de instruções /* ... */

SQL - Data Definition Language (DDL)

- Permite definir várias entidades da BD
- Utilizada para especificar a informação acerca de cada relação:
 - O esquema de cada relação
 - O domínio de valores associados com cada atributo
 - Restrições de integridade (entidade e referencial)
 - O conjunto de índices a manter para cada relação
 - ...
- Notas importantes:
 - Há comandos não disponíveis em alguns SGBD...
 - Devemos consultar o manual do SGBD para uma sintaxe mais completa dos comandos.

Criar e Eliminar uma Base de Dados

- Criar uma base de dados

```
CREATE DATABASE dbname;
```

dbname - nome da base de dados a criar

```
CREATE DATABASE COMPANY;
```

- Eliminar uma base de dados

```
DROP DATABASE dbname;
```

dbname - nome da base de dados a eliminar

```
DROP DATABASE COMPANY;
```

Schema

- Schema é um “namespace” que agrupa tabelas e outros elementos pertencentes à mesma aplicação.
- Criar um Schema

```
CREATE SCHEMA schemaname [AUTHORIZATION username_owner];
```

```
CREATE SCHEMA COMPANY AUTHORIZATION 'CCosta';
```

- Eliminar um Schema

```
DROP SCHEMA schemaname;
```

```
DROP SCHEMA COMPANY;
```

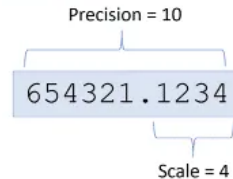
SQL - Tipo de Dados

- Tipos de dados básicos:
 - Numbers
 - Characters, strings
 - Date e time
 - Binary objects
- Os tipos de dados podem variar de acordo com o SGDB!
- Recomendação: Utilizar, na medida do possível, tipos de dados compatíveis com o standard.
 - Aumenta a portabilidade da solução...

SQL - Tipos de dados (SQL:1999)

- Numeric

- NUMERIC(p,s)
- DECIMAL(p,s)
- INTEGER (alias: INT) e.g. 32767
- SMALLINT small integers
- FLOAT(p) e.g. -1E+03
- REAL (for short floats) DOUBLE (for long floats)



- String

- CHARACTER(n) (fixed length)
- CHARACTER (variable length)
- CHARACTER VARYING(n) (alias: VARCHAR(n))
- CLOB (Character Large Object, e.g., for large text)

- Date

- DATE e.g. '1993-01-02'
- TIME e.g. '13:14:15'
- TIMESTAMP e.g. '1993-01-02 13:14:15.000001'

- Binary

- BIT[(n)] e.g. B'01000100'
- BLOB[(n)] e.g. X'49FE' (Binary Large Objects, e.g., for multimedia)

- Boolean

- Boolean

Listagem não exaustiva...

SQL - Tipo de Dados

Alguns mais utilizados...

- **char(n)**
 - cadeia de caracteres de tamanho fixo n
- **varchar(n)**
 - cadeia de caracteres com tamanho máximo n
- **int**
 - números inteiros (4 bytes)
- **numeric(precisão, escala)**
 - números reais “sem limite” de tamanho
- **date e time**
 - data e hora
- **boolean***
 - valores booleanos

* Não existe em SQL Server

SQL Server - Tipos de Dados

Numeric Data Types

Data Type	Description	Length
int	Stores integer values ranging from -2,147,483,648 to 2,147,483,647	4 bytes
tinyint	Stores integer values ranging from 0 to 255	1 byte
smallint	Stores integer values ranging from -32,768 to 32,767	2 bytes
bigint	Stores integer values ranging from -2 ⁵³ to 2 ⁵³ -1	8 bytes
money	Stores monetary values ranging from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
smallmoney	Stores monetary values ranging from -214,748.3648 to 214,748.3647	4 bytes
decimal(p,s)	Stores decimal values of precision <i>p</i> and scale <i>s</i> . The maximum precision is 38 digits	5–17 bytes
numeric(p,s)	Functionally equivalent to decimal	5–17 bytes
float(n)	Stores floating point values with precision of 7 digits (when <i>n</i> =24) or 15 digits (when <i>n</i> =53)	4 bytes (when <i>n</i> =24) or 8 bytes (when <i>n</i> =53)
real	Functionally equivalent to float(24)	4 bytes

SQL Server- Tipos de Dados (cont.)

Character String Data Types

Data Type	Description	Length
<code>char(n)</code>	Stores n characters	n bytes (where n is in the range of 1–8,000)
<code>nchar(n)</code>	Stores n Unicode characters	$2n$ bytes (where n is in the range of 1–4,000)
<code>varchar(n)</code>	Stores approximately n characters	Actual string length +2 bytes (where n is in the range of 1–8,000)
<code>varchar(max)</code>	Stores up to $2^{31}-1$ characters	Actual string length +2 bytes
<code>nvarchar(n)</code>	Stores approximately n characters	$2n$ (actual string length) +2 bytes (where n is in the range of 1–4,000)
<code>nvarchar(max)</code>	Stores up to $((2^{31}-1)/2)-2$ characters	$2n$ (actual string length) +2 bytes

Binary Data Types

Data Type	Description	Length
<code>bit</code>	Stores a single bit of data	1 byte per 8 bit columns in a table
<code>binary(n)</code>	Stores n bytes of binary data	n bytes (where n is in the range of 1–8,000)
<code>varbinary(n)</code>	Stores approximately n bytes of binary data	Actual length +2 bytes (where n is in the range of 1–8,000)
<code>varbinary(max)</code>	Stores up to $2^{31}-1$ bytes of binary data	Actual length +2 bytes

SQL Server- Tipos de Dados (cont.)

Date and Time Data Types

Data Type	Description	Length	Example
date	Stores dates between January 1, 0001, and December 31, 9999	3 bytes	2008-01-15
datetime	Stores dates and times between January 1, 1753, and December 31, 9999, with an accuracy of 3.33 milliseconds	8 bytes	2008-01-15 09:42:16.142
datetime2	Stores date and times between January 1, 0001, and December 31, 9999, with an accuracy of 100 nanoseconds	6–8 bytes	2008-01-15 09:42:16.1420221
datetimeoffset	Stores date and times with the same precision as datetime2 and also includes an offset from Universal Time Coordinated (UTC) (also known as Greenwich Mean Time)	8-10 bytes	2008-01-15 09:42:16.1420221 +05:00
smalldatetime	Stores dates and times between January 1, 1900, and June 6, 2079, with an accuracy of 1 minute (the seconds are always listed as ":00")	4 bytes	2008-01-15 09:42:00
time	Stores times with an accuracy of 100 nanoseconds	3–5 bytes	09:42:16.1420221

Listagem não exaustiva. Há outros tipo como o cursor, sql_variant, table, xml, ...

SQL - Definição de Domínio

- O comando **create domain** permite definir novos tipos de dados.
- Um domain pode conter um valor de defeito (default) e restrições do tipo not null e check.

```
CREATE DOMAIN domainname
```

Criação...

```
CREATE DOMAIN compsalary INTEGER  
    NOT NULL CHECK (compsalary > 750);
```

Utilização...

```
CREATE TABLE EMPLOYEE (  
    ...  
    Salary                compsalary,  
    ...);
```

Nota: Não disponível em SQL SERVER.

SQL - Definição de Novo Tipo

- Como alternativa ao domain, podemos criar só um novo tipo (alias) com o comando **create type**.

CREATE Type... em SQL SERVER

Criação...

```
CREATE TYPE SSN FROM varchar(9) NOT NULL;
```

Utilização...

```
CREATE TABLE EMPLOYEE (  
    ...  
    Ssn SSN,  
    ...);
```

- Nota: Em geral, é mais limitado que o create domain.

DDL - Criar uma Tabela

```
CREATE TABLE tname ( A1 D1, A2 D2, ..., An Dn,  
                      (integrity-constraint1),  
                      ...  
                      (integrity-constraintK) );
```

tname - nome da relação (tabela)

```
CREATE TABLE COMPANY.EMPLOYEE (...)  
CREATE TABLE EMPLOYEE (...)
```

COMPANY - nome do schema

A1 D1, A2 D2, ..., An Dn

A1...An - Atributos da relação

D1...Dn - Domínio dos atributos

Restrições de Integridade

integrity-constraint1,

...,

integrity-constraintN

Criar uma Tabela (exemplo)

CREATE TABLE...
definindo atributos e respectivo domínio.

```
CREATE TABLE EMPLOYEE (  
    Fname          VARCHAR(15),  
    Minit          CHAR,  
    Lname          VARCHAR(15),  
    Ssn            CHAR(9),  
    Bdate          DATE,  
    Address        VARCHAR(30),  
    Sex            CHAR,  
    Salary         DECIMAL(10,2),  
    Super_ssn      CHAR(9),  
    Dno            INT);
```

Atributos - Valores por Omissão

- Podem ser definidos valores por omissão para cada coluna
 - utilizando o termo “default”

CREATE com default ...

```
CREATE TABLE EMPLOYEE (  
    Fname          VARCHAR(15),  
    ...  
    Salary         DECIMAL(10,2)    DEFAULT 0,  
    ...  
    Dno            INT);
```

Restrições de Integridade

- **check** (P)
 - impor uma regra a um atributo
- **not null**
 - atributo não pode ser null
- **primary key** ($A1, \dots, An$)
 - definir chave primária
- **unique** ($A1, \dots, An$)
 - chaves candidatas não primárias
- **foreign key**
 - definir chave estrangeira

As restrições podem ser de:

- **coluna** - referem-se a apenas uma coluna e são descritas em frente à coluna
- **tabela** - referem-se a mais do que a uma coluna e ficam separadas da definição das colunas

Restrição CHECK

Restrição CHECK na coluna...

```
CREATE TABLE EMPLOYEE (  
    ...  
    Salary          DECIMAL(10,2)      CHECK (Salary > 12),  
    ...);
```

Restrição CHECK na tabela...

```
CREATE TABLE DEPARTMENT (  
    ...  
    Dept_create_date    DATE            NOT NULL,  
    Mgr_start_date      DATE,  
    ...  
    CHECK (Dept_create_date <= Mgr_start_date);
```

Restrição aplicada a cada atributo referenciado sempre que um tuplo é introduzido ou modificado.

Restrição PRIMARY KEY

- Só podemos definir uma chave primária na tabela.
 - Por definição, a chave primária não pode conter valores repetidos ou nulos.

Restrição PRIMARY KEY na coluna...

```
CREATE TABLE EMPLOYEE (  
    ...  
    Ssn                CHAR(9)          PRIMARY KEY,  
    ...);
```

Restrição PRIMARY KEY na tabela... (obrigatório se PK for composta por mais do que um atributo)

```
CREATE TABLE EMPLOYEE (  
    ...  
    Ssn                CHAR(9),  
    ...  
    PRIMARY KEY (Ssn));
```


Restrição UNIQUE

- Utilizada para as chaves candidatas alternativas.
 - Não pode conter valores repetidos mas pode ter valores null.

Restrição UNIQUE na coluna...

```
CREATE TABLE DEPARTMENT (  
    Dname                VARCHAR(15)  UNIQUE  NOT NULL,  
    Dnumber              INT           NOT NULL,  
    PRIMARY KEY (Dnumber),  
    ... );
```

Restrição UNIQUE na tabela...

```
CREATE TABLE DEPARTMENT (  
    Dname                VARCHAR(15)           NOT NULL,  
    Dnumber              INT                   NOT NULL,  
    PRIMARY KEY (Dnumber),  
    UNIQUE (Dname), ... );
```

Restrição FOREIGN KEY

- Utilizada para declarar chaves estrangeiras.
- Uma chave estrangeira deve referenciar uma chave primária ou única.

Restrição FOREIGN KEY na coluna...

```
CREATE TABLE EMPLOYEE (  
    ...  
    Super_ssn    CHAR(9)    REFERENCES EMPLOYEE(Ssn),  
    Dno          INT        REFERENCES DEPARTMENT(Dnumber) NOT NULL,  
    ...);
```

Restrição FOREIGN KEY na tabela...

```
CREATE TABLE EMPLOYEE (  
    ...  
    Ssn          CHAR(9),  
    Dno          INT          NOT NULL,  
    ...  
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),  
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
```

Restrição FOREIGN KEY

Integridade Referencial

- Pode haver uma violação quando são inseridos ou eliminados tuplos ou quando os atributos chave estrangeira ou primária são modificados, resultando numa rejeição da operação.
- Podemos definir as seguintes ações alternativas: “on delete” e “on update”, com as seguintes opções:
 - restrict - não deixa efetuar a operação
 - cascade - apaga os registos associados (delete) ou altera a chave estrangeira (update)
 - set null - a chave estrangeira passa a null.
 - set default - a chave estrangeira passa a ter o valor por omissão.

Restrição FOREIGN KEY

Integridade Referencial

Restrição FOREIGN KEY

```
CREATE TABLE EMPLOYEE (  
    ...  
    Ssn          CHAR(9),  
    Dno          INT          NOT NULL,  
    ...  
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)  
        ON DELETE SET NULL ON UPDATE CASCADE,  
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)  
        ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

Se o tuplo do supervisor é eliminado, a coluna Super_ssn dos supervisionados passa automaticamente a Null.

Se o Ssn do supervisor é atualizado, a coluna Super_ssn dos supervisionados é atualizada em cascata.

Restrições - atribuição de nome

- Imaginando que queremos alterar uma restrição de uma tabela... Como referenciá-la?
- Nestas situações temos de “baptizar” a restrição com um nome próprio.

Restrições com nome...

```
CREATE TABLE EMPLOYEE (  
    ...  
    ...  
    CONSTRAINT EMPPK  
        PRIMARY KEY (Ssn),  
    CONSTRAINT EMPSUPERFK  
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)  
            ON DELETE SET NULL ON UPDATE CASCADE,  
    CONSTRAINT EMPDEPTFK  
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)  
            ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

Tabela - Drop

- O comando **drop table** remove da base de dados toda a informação sobre a tabela e os dados (tuplos).

Eliminar a tabela EMPLOYEE

```
DROP TABLE EMPLOYEE;
```

- Caso haja violação de restrições de integridade referencial, a operação é rejeitada.
- No entanto, a opção **CASCADE*** permite eliminar a tabela e os elementos referenciados na restrição.

Eliminar a tabela EMPLOYEE com opção CASCADE

```
DROP TABLE EMPLOYEE CASCADE;
```

* Não está disponível em SQL Server. Solução: eliminar primeiro o constraint.

Tabela - Alter

- O comando **alter table** é utilizado para modificar o esquema da tabela ou restrições existentes.
- Adicionar atributos à tabela:

```
ALTER TABLE tablename ADD Attribute Domain
```

```
ALTER TABLE EMPLOYEE ADD nofiscal INT;
```

- Todos os tuplos existentes ficam com valor null no novo atributo.

- Adicionar restrições à tabela:

```
ALTER TABLE tablename ADD CONSTRAINT name theconstraint
```

```
ALTER TABLE EMPLOYEE ADD CONSTRAINT salarymin CHECK (Salary >475);
```

Tabela - Alter

- Eliminar atributos da tabela:

```
ALTER TABLE tablename DROP COLUMN attributename
```

```
ALTER TABLE EMPLOYEE DROP COLUMN nofiscal;
```

- Eliminar restrições da tabela:

```
ALTER TABLE tablename DROP CONSTRAINT name
```

```
ALTER TABLE EMPLOYEE DROP CONSTRAINT salarymin;
```

- Alterar um atributo de uma tabela:

```
ALTER TABLE tablename ALTER Attribute Domain
```

```
ALTER TABLE EMPLOYEE ALTER COLUMN noFiscall CHAR(9);
```


SQL DDL - Caso de Estudo

Empresa

Esquema Relacional da BD da Empresa

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Empresa - Criação da Relação EMPLOYEE

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation
-------	----------------	-----------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Employee - SQL DDL

```
CREATE TABLE EMPLOYEE (
    Fname          VARCHAR(15)          NOT NULL,
    Minit          CHAR,
    Lname          VARCHAR(15)          NOT NULL,
    Ssn            CHAR(9)              NOT NULL,
    Bdate          DATE,
    Address        VARCHAR(30),
    Sex            CHAR,
    Salary         DECIMAL(10,2),
    Super_ssn      CHAR(9),
    Dno            INT                  NOT NULL,
    PRIMARY KEY (Ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
```

Empresa - Criação da Relação DEPARTMENT

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation
-------	----------------	-----------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Department - SQL DDL

```
CREATE TABLE DEPARTMENT(
    Dname          VARCHAR(15)          NOT NULL,
    Dnumber        INT                   NOT NULL,
    Mgr_ssn        CHAR(9)              NOT NULL,
    Mgr_start_date DATE,
    PRIMARY KEY (Dnumber),
    UNIQUE (Dname),
    FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
```

Empresa - Criação da Relação DEP_LOCATIONS

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation
-------	----------------	-----------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Dept_Locations - SQL DDL

```
CREATE TABLE DEPT_LOCATIONS(
    Dnumber          INT                NOT NULL,
    Dlocation        VARCHAR(15)       NOT NULL,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```

Empresa - Criação da Relação PROJECT

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	M
-------	----------------	---------	---

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Project - SQL DDL

```
CREATE TABLE Project(
    Pname          VARCHAR(15)          NOT NULL,
    Pnumber        INT                  NOT NULL,
    Plocation      VARCHAR(15),
    Dnum           INT                  NOT NULL,
    PRIMARY KEY (Pnumber),
    UNIQUE (Pname),
    FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber));
```

Empresa - Criação da Relação WORKS_ON

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation
-------	----------------	-----------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Works_ON - SQL DDL

```
CREATE TABLE Works_on(
    Essn          CHAR(9)          NOT NULL,
    Pno           INT              NOT NULL,
    Hours         DECIMAL(3,1)     NOT NULL,
    PRIMARY KEY (Essn, Pno),
    FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber));
```

Empresa - Criação da Relação DEPENDENT

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation
-------	----------------	-----------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Dependent - SQL DDL

```
CREATE TABLE Dependent(
    Essn          CHAR(9)          NOT NULL,
    Dependent_name VARCHAR(15)      NOT NULL,
    Sex           CHAR,
    Bdate         DATE,
    Relationship   VARCHAR(8),
    PRIMARY KEY (Essn, Dependent_name),
    FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn));
```


Empresa DDL - Considerações Práticas

EXEMPLO: Employee, Department and Foreign Keys

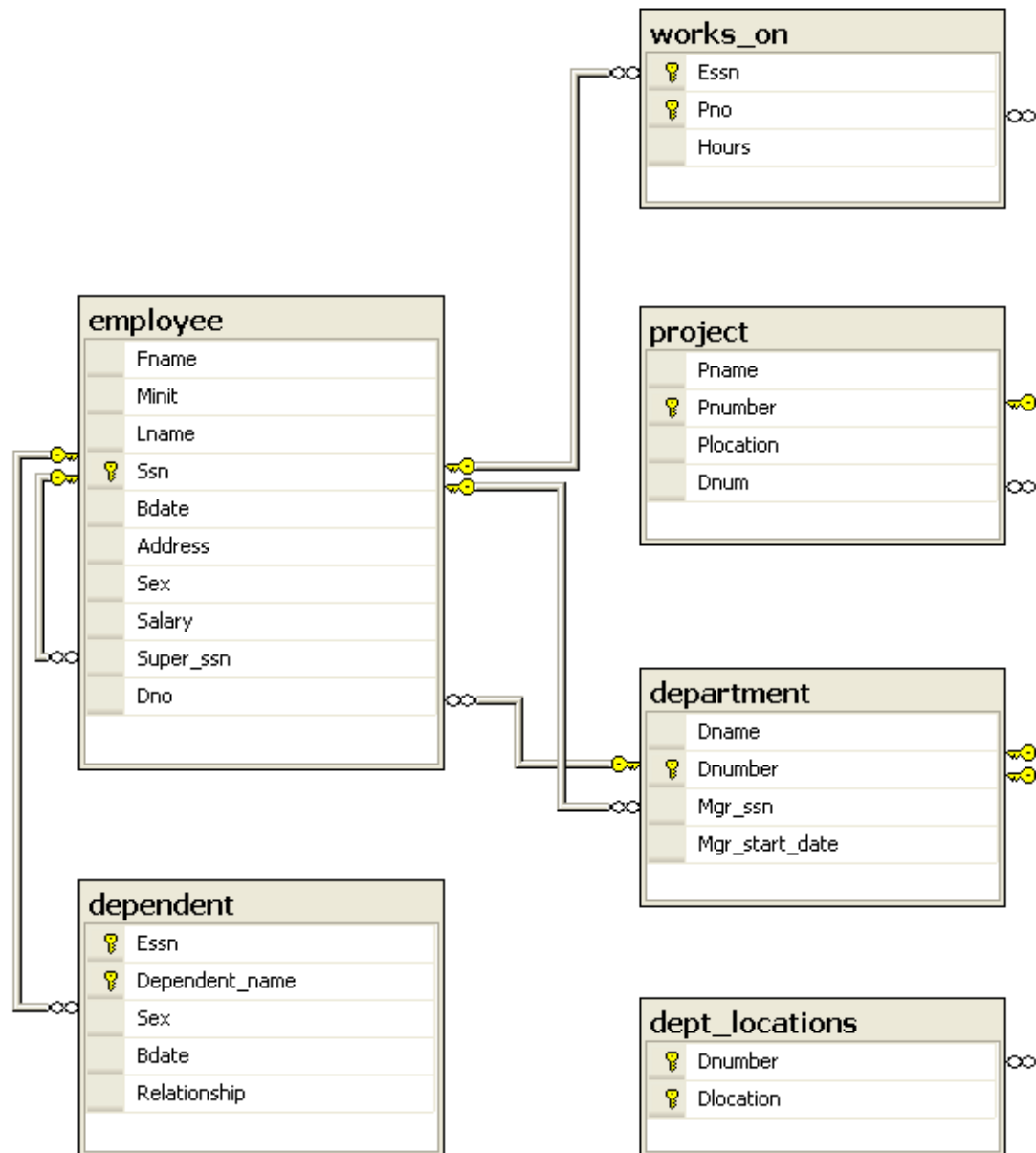
```
CREATE TABLE EMPLOYEE (  
    Ssn          CHAR(9)          NOT NULL,  
    Super_ssn    CHAR(9),  
    Dno          INT              NOT NULL,  
    ...  
    PRIMARY KEY (Ssn),  
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn));
```

```
CREATE TABLE DEPARTMENT(  
    Dnumber      INT              NOT NULL,  
    ...  
    PRIMARY KEY (Dnumber),  
    ...);
```

```
ALTER TABLE EMPLOYEE  
    ADD CONSTRAINT EMPDEPTFK FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber);  
ALTER TABLE DEPARTMENT  
    ADD CONSTRAINT DEPTMGRFK FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn);
```

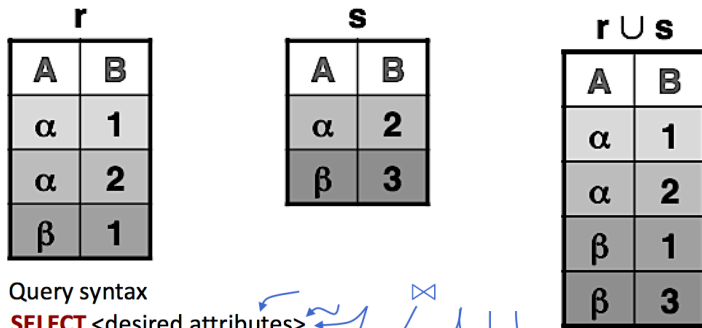
- Na prática só podemos criar restrições de integridade referencial, com recurso a chaves estrangeiras, quando temos as duas relações criadas.
- Assim, devemos começar por criar cada uma das relações (tabelas) e só depois definir as restrições.
 - Ou pelo menos uma delas...

SQL Server - Database Diagram



A Seguir?

Data Operations – Relational Algebra



Query syntax

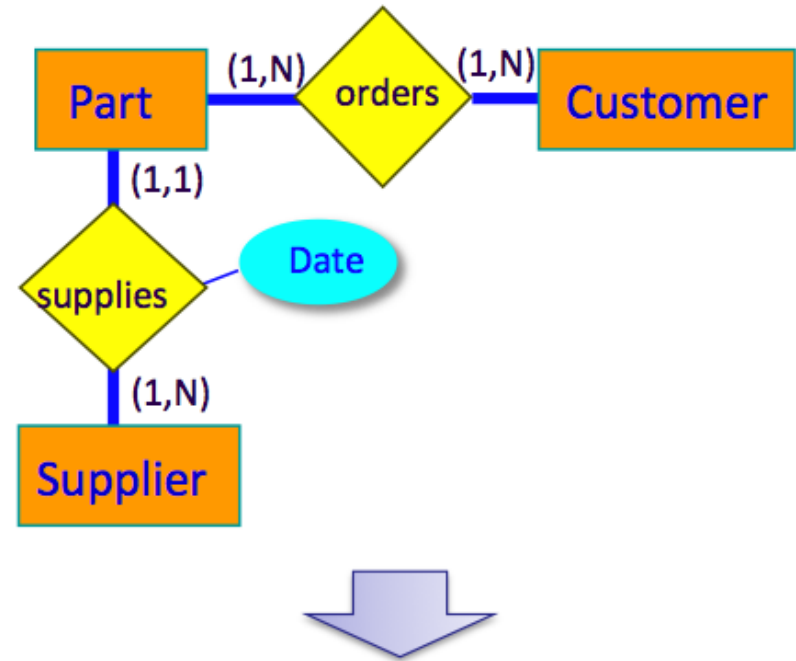
SELECT <desired attributes>
FROM <one or more tables>
WHERE <predicate holds for selected tuple>
GROUP BY <key columns, aggregations>
HAVING <predicate holds for selected group>
ORDER BY <columns to sort>



SQL – Describe Database Schema

```
CREATE TABLE DEPARTMENT
( Dname          VARCHAR(15)    NOT NULL,
  Dnumber        INT            NOT NULL,
  Mgr_ssn        CHAR(9)       NOT NULL,
  Mgr_start_date DATE,
  PRIMARY KEY (Dnumber),
  UNIQUE (Dname),
  FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
```

The E/R Model (Conceptual Model)



The Relational Schema

Part (Name,Description,Part#)
Supplier (Name, Addr)
Customer (Name, Addr)
Supplies (Name,Part#, Date)
Orders (Name,Part#)



Resumo

- Introdução ao SQL
- SQL DDL
 - Definição de dados (relações)
 - Definição de tipos de dados
 - Definição de restrições de integridade
- Caso de Estudo