

Bases de Dados

Pedro Furtado

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia da Universidade de Coimbra
2019->

Integrity in the Relational Model

What is integrity and consistency?

What are they, what are they for and how do they guarantee themselves?

What's with integrity rules?

The problem of banks and other institutions in the 1980s and 1990s

duplicate customer records (multiplied)

many records with a lot of empty information

O que evitar?

- **Non-elementary information (multiple descriptive attributes) should not be repeated in the database...**
- **There should be no inconsistent information in the DB**
-

Objective = Ensure integrity, consistency of data

RULES OF INTEGRITY AND CONSISTENCY

Integrity rules: summary

- **Integridade de domínio** (domain integrity)

Forces attribute values to be within certain values, are required, etc.

Integridade de entidade (entity integrity)

Primary key definition for each table. Forces that there are no records in a table with the same values in the attributes of the primary key, allowing you to individually identify all records in the table.

Integridade referencial (referential integrity)

Definition of foreign key. Allows you to correctly relate the records of two tables, ensuring that the foreign key values of each record exist forcibly in the other table (in the table where the foreign key corresponds to the primary key).

Regras complexas de negócio (complex business logic)

More complex business rules that cannot be defined in the structure of tables and that are checked through programs.

Integrity restrictions

Integrity restrictions are rules that define what data is valid. Let's look at some examples for the Teacher table:

Wages must be greater than or equal to zero (cannot be negative)

The salary has to be less than \$5,000

The date of birth must be between 01-01-1900 and the day on which the registration is entered

The Name field cannot be null (fill required)

The BI field must be unique (there can be no duplications)

These rules are part of the definition of the table structure (they are usually indicated when defining the table), others are checked in the applications

The Database Management System (DBMS) verifies that the data respects the rules defined each time operations that change the data (insert, update, or delete records) are performed

Preserving data integrity

- **Integrity constraints defined**
 - na criação de tabelas;
 - na alteração de tabelas.
- **Integrity restrictions verified in the execution of**
 - INSERT, UPDATE, DELETE.

Parâmetros das restrições

- **CONSTRAINT <nome>** Especifica nome para a restrição. Útil em caso de erro. É opcional.
- **NULL / NOT NULL** Especifica que uma coluna (ou grupo de colunas) não possa(m) conter valores nulos
- **UNIQUE** Especifica que uma coluna (ou grupo de colunas) só pode(m) conter valores unívocos
- **PRIMARY KEY** Especifica que uma coluna (ou grupo de colunas) forma(m) uma chave primária
- **FOREIGN KEY (coluna,...)
REFERENCES TABLE (coluna,...)** Especifica que uma coluna (ou grupo de colunas) são uma chave externa
- **CHECK** Especifica uma condição a verificar para que o registo possa existir na tabela

UC

bd

Tipos de restrições

- **Restrição de tabela** (aplica-se a uma ou mais colunas)

Ex:

```
CREATE TABLE DISCIPL  
(NOME CHAR(60), NOTA NUMBER(2), EPOCA CHAR(8),  
CONSTRAINT DISCIPL_CP PRIMARY KEY (NOME, NOTA, EPOCA) );
```

- **Restrição de coluna** (aplica-se só a uma coluna)

Ex:

```
CREATE TABLE ELEITOR  
(NOME CHAR(60) PRIMARY KEY  
....);
```


Primary, foreign keys, constraints, SQL

In the relational model ...

A primary key is required in each table

Patient (Person)

numCC	First Name	Middle Initial	Last Name	Address	City	Zip Code	Country	Phone Number	Date of Birth
19845	Joaquim	N.	Gomes	Rua Castilho, 23	Lisboa	2720-234	Portugal	91-99893	2/10/00
21537	Manuel	M.	Silva	Rua Augusta, 18	Coimbra	3000-542	Portugal	96-39494	2/10/90
82435	Victor	P.	Correira	Rua da Prata	Lisboa	2453-122	Portugal	21-789876	1/9/87

```
CREATE TABLE Person(
  numCC integer PRIMARY KEY,
  FirstName varchar(50),
  MiddleInitial varchar(50),
  LastName varchar(50),
  Address varchar(100),
  City varchar(50),
  ZipCode varchar(10),
  Country String varchar(50),
  PhoneNumber varchar(14),
  DateOfBirth Date);
```

- What's it for?
- Why is it important?
- What does that entail?

Patient (Person)

numCC	First Name	Middle Initial	Last Name	Address	City	Zip Code	Country	Phone Number	Date of Birth
19845	Joaquim	N.	Gomes	Rua Castilho, 23	Lisboa	2720-234	Portugal	91-99893	2/10/00
21537	Manuel	M.	Silva	Rua Augusta, 18	Coimbra	3000-542	Portugal	96-39494	2/10/90
82435	Victor	P.	Correira	Rua da Prata	Lisboa	2453-122	Portugal	21-789876	1/9/87

- What's it for?

To uniquely identify each row in the table

- Why is it important?

Because it allows access to a specific row directly in many operations

It also reinforces the logic of uniqueness (a line does not repeat)

UC

bd

```
CREATE TABLE Person(
  numCC integer PRIMARY KEY,
  FirstName varchar(50),
  MiddleInitial varchar(50),
  LastName varchar(50),
  Address varchar(100),
  City varchar(50),
  ZipCode varchar(10),
  Country String varchar(50),
  PhoneNumber varchar(14),
  DateOfBirth Date);
```

Chave primária

- Primary key (or identifier): attribute, or set of attributes that allow you to unequivocally identify any record of a table. The primary key is:
- **Unambiguous:** The attribute (or attributes) of the primary key have a single value for any record in the table.
- **Non-null:** There can be no records in the table that have the attribute (or attributes) of the null primary key (with nothing).

The BI attribute can be the primary key of the Cliente_banco

Cliente_banco

Nome	Cidade	BI	Data_nascimento	N_de_conta
António Silva	Coimbra	1234343	12-03-1968	900
Joaquim Alves Dias	Guarda	1256673	18-01-1978	556
Maria Teresa Horta	Lisboa	1275432	15-09-1978	900
Joana Antunes	Lisboa	1022634	23-07-1988	801
Luísa Saraiva	Lisboa	1342664	22-04-1989	647

- The properties of the primary key (**uniqueness and non-null**) are **constraints** automatically enforced by the DBMS, which means:
 - if you try to insert a new row with a null value or with a **value that already exists in the table for that primary key attribute, it will not be inserted and it will raise an error.**

```
CREATE TABLE Person(  
  numCC integer PRIMARY KEY,  
  FirstName varchar(50),  
  MiddleInitial varchar(50),  
  LastName varchar(50),  
  Address varchar(100),  
  City varchar(50),  
  ZipCode varchar(10),  
  Country String varchar(50),  
  PhoneNumber varchar(14),  
  DateOfBirth Date);
```

PK is a constraint... And there are many constraints...

- **Concept of Constraint:** the concept of a constraint is a restriction on the values that an attribute can take in a specific context based on some definition of the constraint.
- Constraints are **automatically enforced by the DBMS**, which means that, if a constraint is violated during some operation, then the operation aborts and an **error is raised**.

Candidate Primary Keys

- A **candidate primary key** is an **attribute** or a **minimum set of attributes** that are **unique and non-null** in the **context of the application**
- In this context a **minimum set** means a **set of attributes that is not a superset of a smaller set of attributes that are also unique and non-null**.

```
CREATE TABLE Person(  
  numCC integer PRIMARY KEY,  
  FirstName varchar(50),  
  MiddleInitial varchar(50),  
  LastName varchar(50),  
  Address varchar(100),  
  City varchar(50),  
  ZipCode varchar(10),  
  Country String varchar(50),  
  PhoneNumber varchar(14),  
  DateOfBirth Date);
```

```
Create table registration(  
  studentID integer,  
  courseID integer,  
  primary key(studentID, courseID) );
```


Still on the key candidate primaries

- There are often several possibilities to form the primary key of a table. Each of the attributes or sets of attributes that can constitute a primary key is called the candidate key.
- **Only one of the candidate keys can be declared as a primary key.**
- If the table has no candidate key, a numeric attribute is introduced that is only for making the key role = artificial key

Matriculas

N_matricula	Nome_aluno	Numero	Disciplina	Data_matricula
100	António Silva	1234343	Português	12-03-2004
101	António Silva	1234343	Análise	12-03-2004
102	Joana Antunes	1275432	Álgebra	13-03-2004
103	Joana Antunes	1275432	Português	13-03-2004
104	Luísa Saraiva	1342664	Análise	13-03-2004

**Chaves candidatas
serão por exemplo?**

{N_matricula}
{Numero, Disciplina}
{Nome_aluno, Disciplina}

E se poder chumbar/repetir?

Matriculas

N_matricula	Nome_aluno	Numero	Disciplina	Data_matricula
100	António Silva	1234343	Português	12-03-2004
101	António Silva	1234343	Análise	12-03-2004
102	Joana Antunes	1275432	Álgebra	13-03-2004
103	Joana Antunes	1275432	Português	13-03-2004
104	Luísa Saraiva	1342664	Análise	13-03-2004

Chaves candidatas:

{N_matricula}
{Numero, Disciplina}
{Nome_aluno, Disciplina}

- What if a person enrolls two years later, for having failed the discipline?
- What matters is not so much what's on the table at any given time, but what can be inserted...

{N_matricula}
{Numero, Disciplina, **Data_matricula**}
{Nome_aluno, Disciplina, **Data_matricula**}

- ***"In this context a **minimum set** means a set of attributes that is not a superset of a smaller set of attributes that are also unique and non-null."***



{N_matricula}

{Numero, Disciplina, Data_matricula}

{Nome_aluno, Disciplina, Data_matricula}

Artificial keys

- The primary key needs to be **unique** and **non-null**
- **It is possible to always meet these two requirements by simply creating an ID** field for every table that is filled by a counter and define it as the primary key.
- A counter is a non-volatile variable that gives you the next value of a sequence that is unique and non-null everytime you request the new value: 1,2,3,4,5,...,
- When you insert a new row into the table, you use the next value in the sequence as the primary key (the ID).

PersonID int NOT NULL AUTO_INCREMENT Primary key

PersonID int NOT NULL serial Primary key

Let's create table and insert data with serial

```
CREATE TABLE books (  
  bookID SERIAL PRIMARY KEY,  
  title   VARCHAR(200) NOT NULL,  
  author  VARCHAR(200) NULL  
);
```

Figure 23. Serial

Inserting into books should use either the DEFAULT keyword or omit the ID column from the INSERT list:

```
INSERT INTO books (bookID, title, author)  
VALUES (DEFAULT, '1984', 'George Orwell');
```

```
INSERT INTO books (bookID, title, author)  
VALUES ( 'Animal Farm', 'George Orwell');
```

Figure 24. Inserting into serial

Sequence: besides the autoincrement, serial and identity datatypes, there is also the sequence construct. Sequence is part of the SQL standard and very easy to use (postgres):

```
create sequence bookID;
```

```
insert into books
```

```
VALUES(nextval('bookID'),'1984','Geaorge Orwell');
```

Figure 27. Sequence in postgres

```
CREATE [ TEMPORARY | TEMP ] SEQUENCE [ IF NOT EXISTS ] name  
  [ AS data_type ]  
  [ INCREMENT [ BY ] increment ]  
  [ MINVALUE minvalue ] [ MAXVALUE maxvalue ]  
  [ START [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE ]  
  [ OWNED BY { table_name.column_name | NONE } ]
```

Figure 28. Create Sequence options in postgres

Should we use **artificial** or **natural** keys?

- Some developers always create such an autoincrement primary key for every table
- No semantic value: e.g. ID 45 does not have any meaning to the application except that it is a unique unrepeated value
- Persons have an ID card with its own unique and non-null numeric identifier that can be used as primary key (a natural primary key). Then, why create a totally new artificial attribute in that case?

```
Student(ID, studentNr, name, dateOfBirth, IDclass)  
Class(ID, classNr, teacher)
```

```
Student(studentNr, name, dateOfBirth, classNr)  
Class(classNr, teacher)
```

Figure 30. Create Table with artificial ID versus natural key

- In some development/programming environments such as django, it may be simpler (django case) or even mandatory (not the case of django) to always use artificial PKs(IDs)

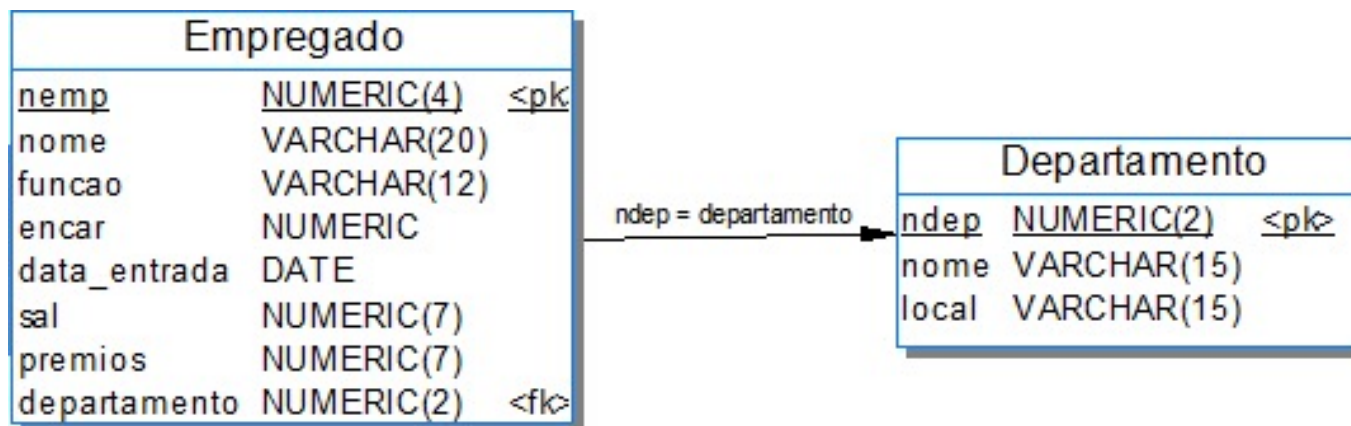
-

Chave estrangeira

Foreign key

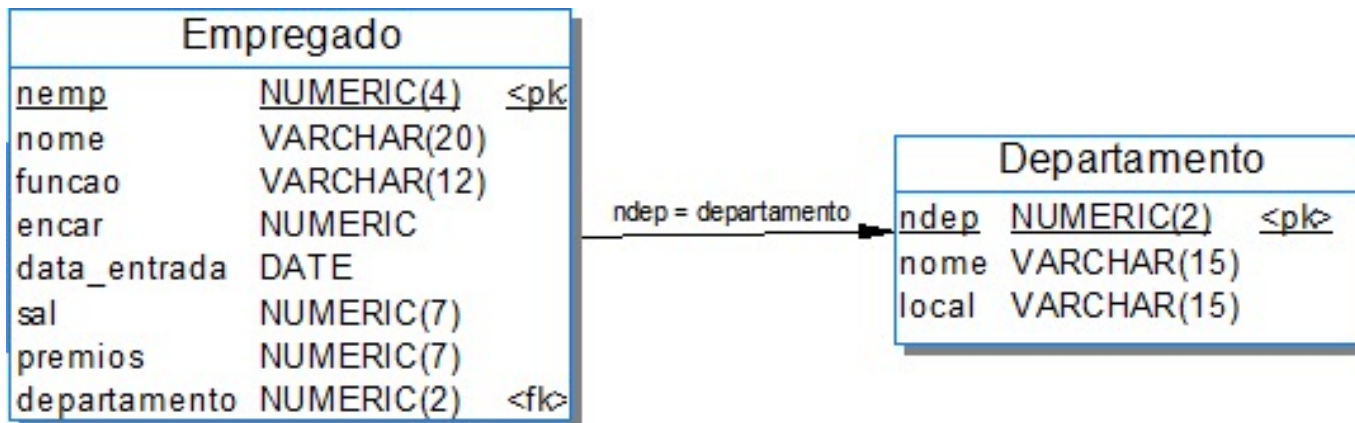
Foreign key => integrity

- I can't enter/delete/modify values that violate relationship.
What does that mean in this?



nemp	nome	funcao	encar	data_entrada	sal	premios	ndep
1839	Jorge Sampaio	Presidente		11-02-1984	890000		10
1566	Augusto Reis	Encarregado	1839	13-02-1985	450975		20
1698	Duarte Guedes	Encarregado	1839	25-11-1991	380850		30
1782	Silvia Teles	Encarregado	1839	03-11-1986	279450		10
1788	Maria Dias	Analista	1566	07-11-1982	565000		20
1902	Catarina Silva	Analista	1566	13-04-1993	435000		20
1499	Joana Mendes	Vendedor	1698	04-10-1984	145600	56300	30

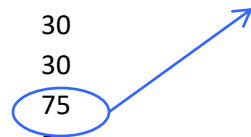
Referential Integrity Issue?



nemp	nome	funcao	encar	data_entrada	sal	premios	ndep
1839	Jorge Sampaio	Presidente		11-02-1984	890000		10
1566	Augusto Reis	Encarregado	1839	13-02-1985	450975		20
1698	Duarte Guedes	Encarregado	1839	25-11-1991	380850		30
1782	Silvia Teles	Encarregado	1839	03-11-1986	279450		10
1788	Maria Dias	Analista	1566	07-11-1982	565000		20
1902	Catarina Silva	Analista	1566	13-04-1993	435000		20
1499	Joana Mendes	Vendedor	1698	04-10-1984	145600	56300	30
1521	Nelson Neves	Vendedor	1698	27-02-1983	212250	98500	30
1654	Ana Rodrigues	Vendedor	1698	17-12-1990	221250	81400	30
1844	Manuel Madeira	Vendedor	1698	21-04-1985	157800	0	30
1900	Tome Ribeiro	Continuo	1698	05-03-1994	56950		75
1876	Rita Pereira	Continuo	1788	07-02-1996	65100		20
1934	Olga Costa	Continuo	1782	22-06-1986	68300		10
1369	Antonio Silva	Continuo	1902	22-12-1996	70800		20

ndep	nome	local
10	Contabilidade	Condeixa
20	Investigação	Mealhada
30	Vendas	Coimbra
40	Planeamento	Montemor

?



Chave estrangeira

Foreign key: Attribute or set of attributes of a table that appears as the primary key in another table, allowing you to establish the relationship between records of those tables.

Cliente_banco


 BI	Nome	Cidade	Data_nascimento	N_de_conta
1234343	António Silva	Coimbra	12-03-1968	900
1256673	Joaquim Alves Dias	Guarda	18-01-1978	556
1275432	Maria Teresa Horta	Lisboa	15-09-1978	900
1022634	Joana Antunes	Lisboa	23-07-1988	801
1342664	Luísa Saraiva	Lisboa	22-04-1989	647

N_de_conta é
chave estrangeira
na tabela
Cliente_banco

BI é chave primária
nesta tabela
Cliente_banco

N_de_conta é
chave primária
na tabela Conta

Conta

 N_de_conta	Saldo	Data_abertura
900	55	12-03-2002
556	1 000	18-01-2002
647	15 336	15-09-2003
801	533	23-07-2003

Foreign Key (owner of apartment?)

```
CREATE TABLE Person(
  numIC integer PRIMARY KEY,
  FirstName varchar(50),
  MiddleInitial varchar(50),
  LastName varchar(50),
  Address varchar(100),
  City varchar(50),
  ZipCode varchar(10),
  Country String varchar(50),
  PhoneNumber varchar(14),
  DateOfBirth Date);
```

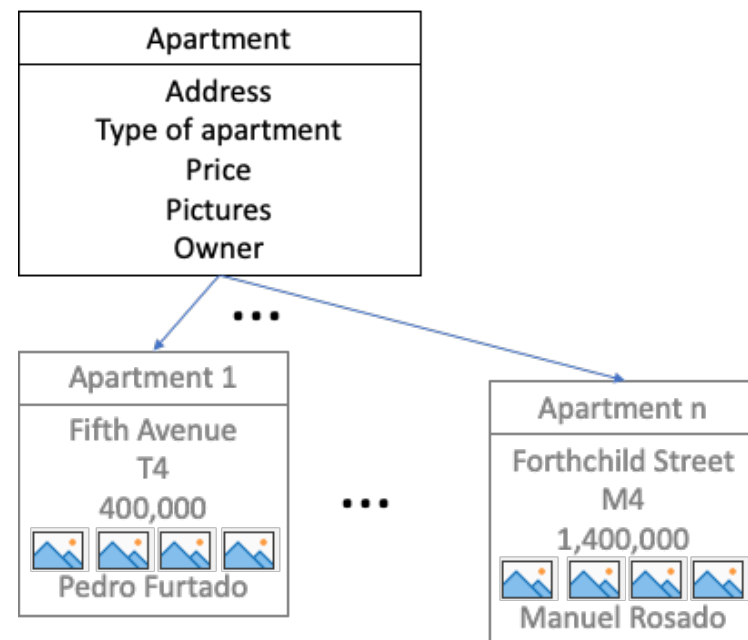


Figure 20. Create Table with Primary key (PK) constraint

FOREIGN KEY (FK)

```
CREATE TABLE Apartment (
  apartmentID          integer PRIMARY KEY,
  Address  varchar(100) not null,
  City      varchar(100) not null,
  Zipcode  varchar(12) not null,
  Type      varchar(5) not null,
  Price     integer,
  creationDate  Date,
  Owner      integer Foreign Key references Person(numIC)
);
```

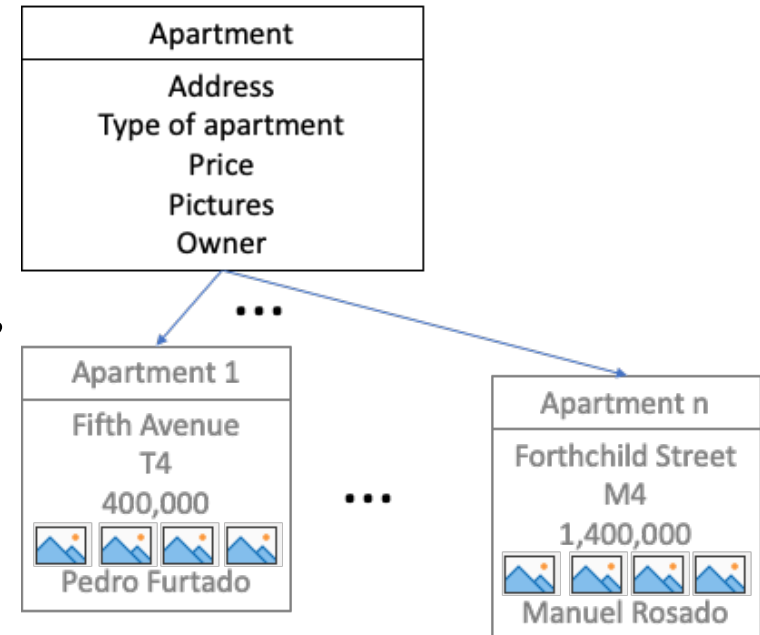
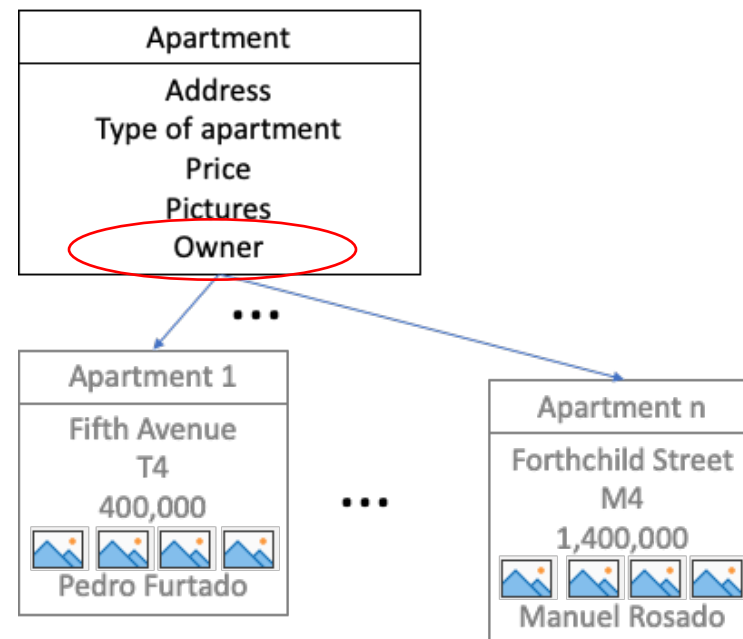


Figure 31. Create Table with Foreign key constraint (also PK)

Foreign key is a constraint. What does it constrain?

- The foreign key constraint is automatically enforced by the DBMS
- there can be no foreign key value that does not exist in the attribute of the referenced table
- the value of **owner** attribute in Apartment **must exist as numIC in the Person table**



Owner integer **Foreign Key references Person(numIC)**

Foreign key in relationship M:M

A bank account can be owned by multiple persons

A person can have multiple bank accounts

Cliente – conta

If I want to add account 556 to Antonio Silva?

Cliente_banco

Nome	Cidade	BI	Data_nascimento	N_de_conta
António Silva	Coimbra	1234343	12-03-1968	900
Joaquim Alves Dias	Guarda	1256673	18-01-1978	556
Maria Teresa Horta	Lisboa	1275432	15-09-1978	900
Joana Antunes	Lisboa	1022634	23-07-1988	801
Luísa Saraiva	Lisboa	1342664	22-04-1989	647

N_de_conta é
chave estrangeira
na tabela
Cliente_banco

UC

Conta

 N_de_conta	Saldo	Data_abertura
900	55	12-03-2002
556	1 000	18-01-2002
647	15 336	15-09-2003
801	533	23-07-2003

bd

ERRADO...

relational model does not allow...
how would it be to search?

Cliente_banco


 BI	Nome	Cidade	Data_nascimento	N_de_conta
1234343	António Silva	Coimbra	12-03-1968	900, 556
1256673	Joaquim Alves Dias	Guarda	18-01-1978	556
1275432	Maria Teresa Horta	Lisboa	15-09-1978	900
1022634	Joana Antunes	Lisboa	23-07-1988	801
1342664	Luísa Saraiva	Lisboa	22-04-1989	647

N_de_conta é
chave estrangeira
 na tabela
 Cliente_banco

BI é chave primária
 nesta tabela
 Cliente_banco

N_de_conta é
 chave primária
 na tabela Conta

Conta

 N_de_conta	Saldo	Data_abertura
900	55	12-03-2002
556	1 000	18-01-2002
647	15 336	15-09-2003
801	533	23-07-2003

UC

bd

ERRADO...

Multiple fields for the same attribute is also non-acceptable....

How would it be to Search?


Cliente_banco

 BI	Nome	Cidade	Data_nascimento	N_de_conta 1	N_de_conta 2
1234343	António Silva	Coimbra	12-03-1968	900	556
1256673	Joaquim Alves Dias	Guarda	18-01-1978	556	
1275432	Maria Teresa Horta	Lisboa	15-09-1978	900	
1022634	Joana Antunes	Lisboa	23-07-1988	801	
1342664	Luísa Saraiva	Lisboa	22-04-1989	647	

BI é chave primária
nesta tabela
Cliente_banco


N_de_conta é
chave primária
na tabela Conta

Conta


 N_de_conta	Saldo	Data_abertura
900	55	12-03-2002
556	1 000	18-01-2002
647	15 336	15-09-2003
801	533	23-07-2003

Foreign key on M:N

Cliente_banco

 BI	Nome	Cidade	Data_nascimento
1234343	António Silva	Coimbra	12-03-1968
1256673	Joaquim Alves Dias	Guarda	18-01-1978
1275432	Maria Teresa Horta	Lisboa	15-09-1978
1022634	Joana Antunes	Lisboa	23-07-1988
1342664	Luísa Saraiva	Lisboa	22-04-1989

ContaCliente

 N_de_conta	BI
900	1234343
900	1275432
556	1234343
...	...

Conta

 N_de_conta	Saldo	Data_abertura
900	55	12-03-2002
556	1 000	18-01-2002
647	15 336	15-09-2003
801	533	23-07-2003


Foreign key

create table ContaCliente(
 N_de_conta **references** conta(N_de_conta),
 BI **references** Cliente_banco(BI),
 primary key(bi,N_de_conta));

Cliente_banco

 BI	Nome	Cidade	Data_nascimento
1234343	António Silva	Coimbra	12-03-1968
1256673	Joaquim Alves Dias	Guarda	18-01-1978
1275432	Maria Teresa Horta	Lisboa	15-09-1978
1022634	Joana Antunes	Lisboa	23-07-1988
1342664	Luísa Saraiva	Lisboa	22-04-1989

ContaCliente

 N_de_conta	BI
900	1234343
900	1275432
556	1234343
...	...


Conta

 N_de_conta	Saldo	Data_abertura
900	55	12-03-2002
556	1 000	18-01-2002
647	15 336	15-09-2003
801	533	23-07-2003


Foreign key

```
create table ContaCliente(
    N_de_conta integer references conta(N_de_conta),
    BI integer references Cliente_banco(BI),
    primary key(BI,N_de_conta) );
```


Cliente_banco

 BI	Nome	Cidade	Data_nascimento
1234343	António Silva	Coimbra	12-03-1968
1256673	Joaquim Alves Dias	Guarda	18-01-1978
1275432	Maria Teresa Horta	Lisboa	15-09-1978
1022634	Joana Antunes	Lisboa	23-07-1988
1342664	Luísa Saraiva	Lisboa	22-04-1989

ContaCliente

 N_de_conta	BI
900	1234343
900	1275432
556	1234343
...	...

Conta

 N_de_conta	Saldo	Data_abertura
900	55	12-03-2002
556	1 000	18-01-2002
647	15 336	15-09-2003
801	533	23-07-2003

```
create table ContaCliente(
    N_de_conta integer,
    BI integer,
    foreign key N_de_conta references conta(N_de_conta),
    foreign key BI references Cliente_banco(BI),
    primary key(BI,N_de_conta) );
```



Create table **Customer** (
numIC integer **PRIMARY KEY**,
 Name varchar(100) not null,
 City varchar(100) not null,
 Birth_date date);

Create table **Account** (
Account_num integer **PRIMARY KEY**,
 Balance numeric(10,2) not null,
 Creation_date date not null);


Create table **Customer_Account** (
numIC integer,
Account_num integer,
PRIMARY KEY(numIC, Account_num),
FOREIGN KEY(numIC) references Customer(numIC),
FOREIGN KEY(Account_num) references Account(Account_num)
);

Figure 37. Table creation statements for bank accounts


Cliente_banco

 BI	Nome	Cidade	Data_nascimento
1234343	António Silva	Coimbra	12-03-1968
1256673	Joaquim Alves Dias	Guarda	18-01-1978
1275432	Maria Teresa Horta	Lisboa	15-09-1978
1022634	Joana Antunes	Lisboa	23-07-1988
1342664	Luisa Saraiva	Lisboa	22-04-1989

ContaCliente

 N_de_conta	BI
900	1234343
900	1275432
556	1234343
...	...

Conta

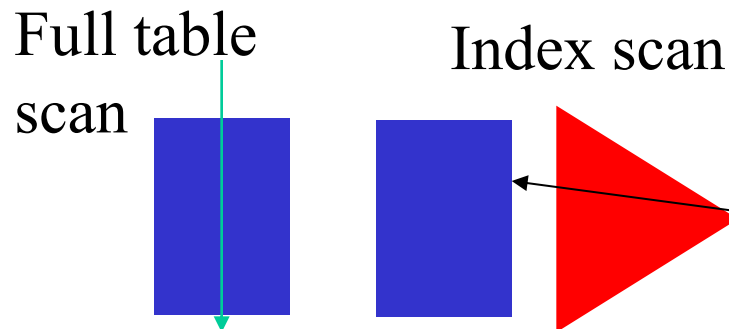
 N_de_conta	Saldo	Data_abertura
900	55	12-03-2002
556	1 000	18-01-2002
647	15 336	15-09-2003
801	533	23-07-2003

Primary keys also have indexes

???

primary key -> indice

- Created automatically when you create the table
- Updated automatically every time I enter the line



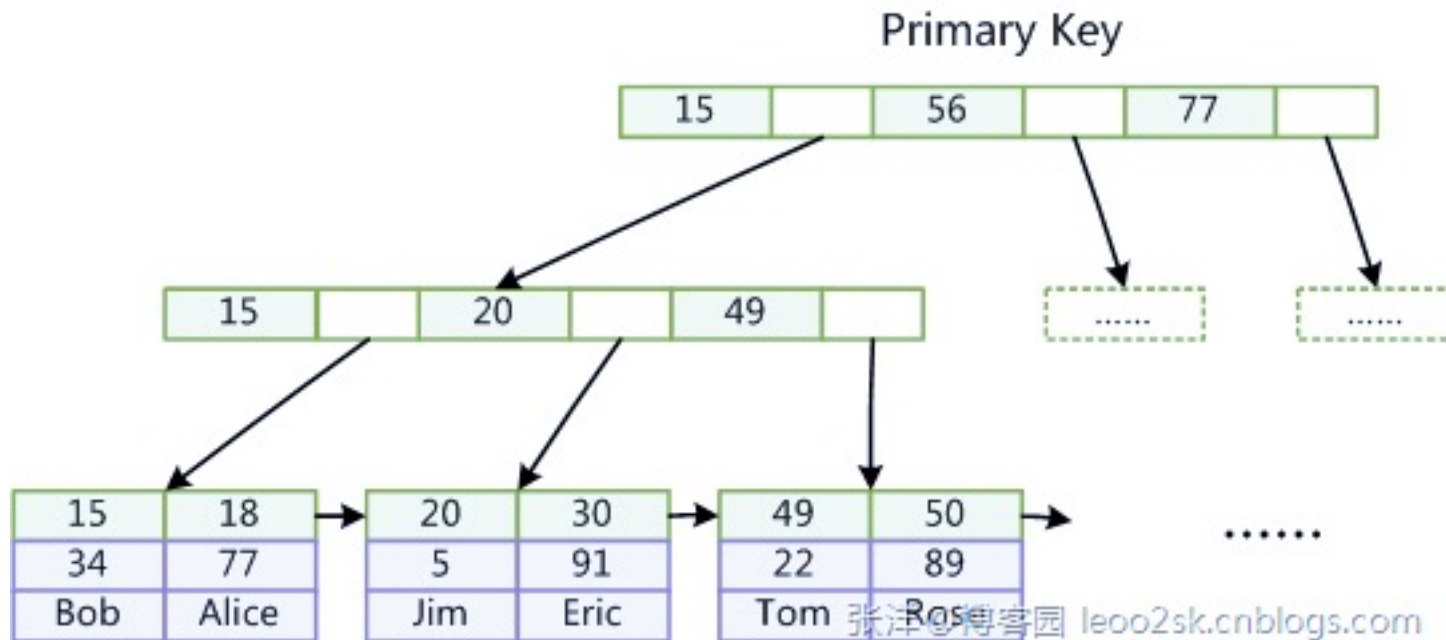
Also, one can create any other index manually:

```
create index idxData on emp(dataEntrada);
```

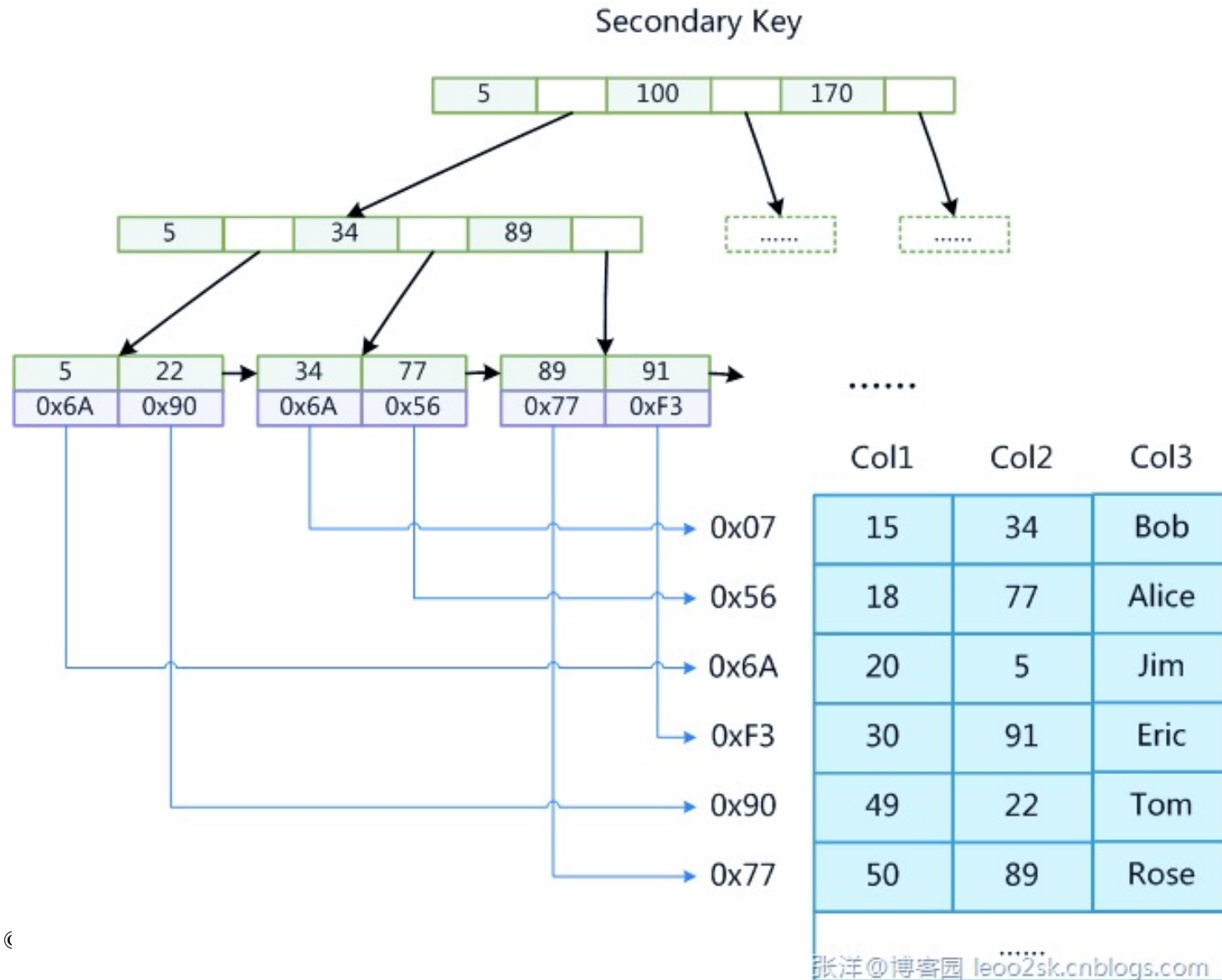
Indexes are ...

- Advantageous for searching specific values or short intervals...
- Not useful for getting a lot of data...
- Disadvantageous in terms of insert/update/delete operations (because index also has to be updated) – e.g. instead of 1ms op takes 2 to 50 ms, varied and depending on factors
- More indexes on table -> more time to insert/update/delete

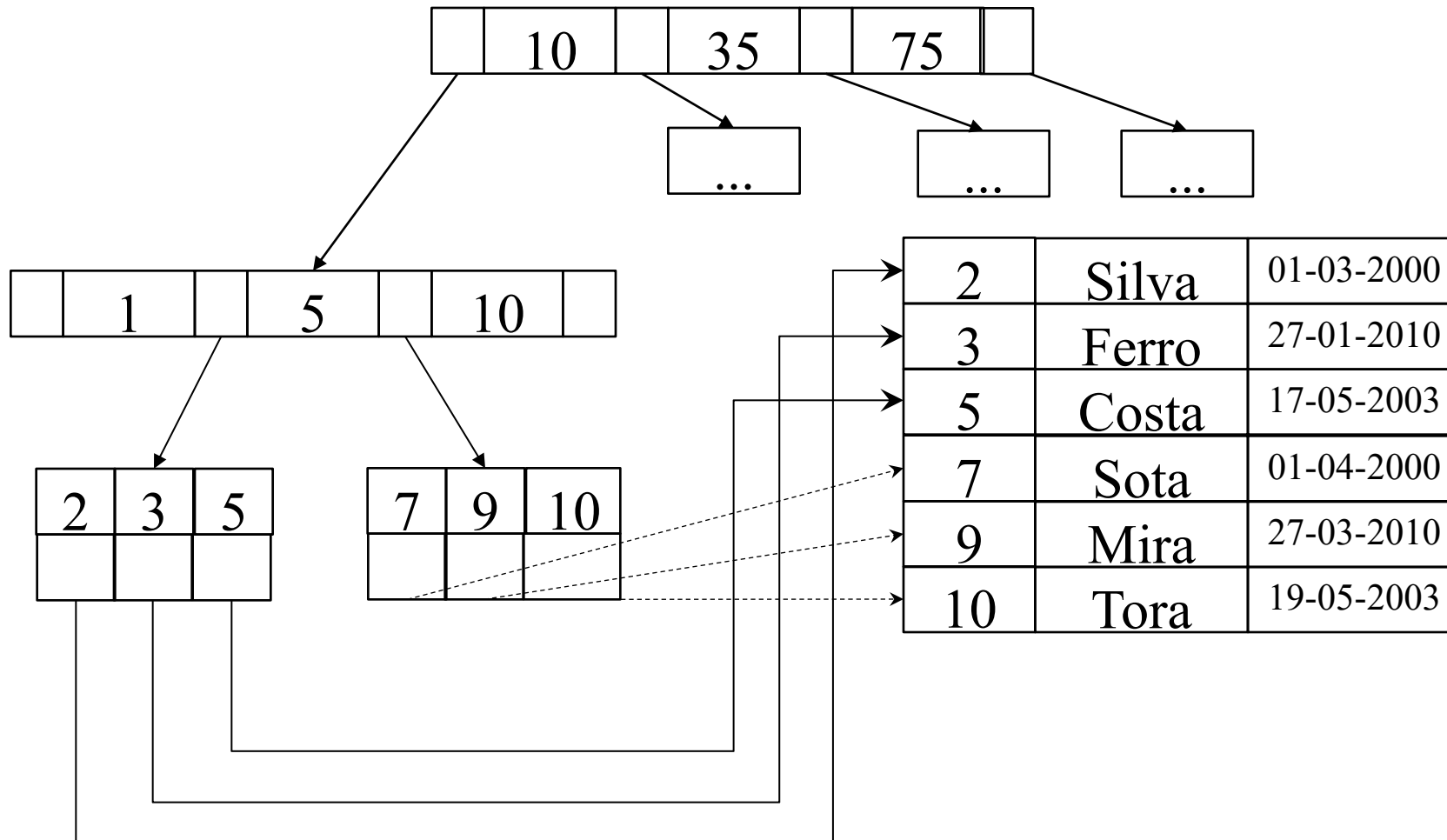
Logical model of an index...



Logical model of an index...



Modelo lógico de um índice...



Other constraints ...

What are they?

Integridade de domínio

Força a que os valores de atributos estejam dentro de certos valores, sejam de preenchimento obrigatório, etc

Tipos de dados (postgres)

Examples of Other types	observations
numeric(n)	subset of integers
numeric(n,p) or decimal(n,p)	decimal number, with n digits and precision p, a subset of real
float(p)	decimal number, with precision p
char(n)	fixed size string
varchar(n)	variable size string
text	text up to 65535 chars
longtext	text up to 4G chars
Blob	binary large object
long blob	long binary large object
date	
datetime	date and time
time	
timestamp	
year	

	Integer numbers	
Storage	Name	Range
Numeric 1 byte	tinyint(n)	-127 to 127
Numeric 2 bytes	smallint(n)	-32768 to 32768
...
Numeric 4 bytes	int(n)	+/-2.147E+9
Numeric 8 bytes	bigint(n)	+/-9.22E+18

- Que tipo de dados para cartão de cidadão?

8085771 4xz5 -> 08085771 4xz5

char(8),char(12), varchar(8), varchar(12)

integer(8), numeric(8)? + char(4)?

faço contas?

- porque é que datas têm um tipo de dados específico?
- Que tipo de dados para videos, documentos, etc?
 - 2 hipóteses: BLOB ou link em varchar

Com regras e check...

- Os salário tem de ser maior ou igual a zero (não pode ser negativo)
- O salário tem de ser menor do que 5 mil dólares

```
create table emp(..., sal numeric(7) check(sal >= 0 and sal < 5000) ...);
```

- A data de nascimento tem de estar compreendida entre 01-01-1900 e o dia em que se está a inserir o registo

```
create table emp(...,  
    dataNasc check(dataNasc between '01.01.1900' and now()) ,  
    ...  
);
```

- O campo Nome não pode ser nulo (preenchimento obrigatório)

```
create table emp(..., name varchar(20) not null ...);
```

CHECK

```
CREATE TABLE Person(  
  IdentityCardID Numeric(12) PRIMARY KEY,  
  FirstName varchar(50) NOT NULL,  
  MiddleInitial varchar(50) NULL,  
  LastName varchar(50) NOT NULL,  
  Address varchar(100) NOT NULL,  
  City varchar(50) NOT NULL,  
  ZipCode varchar(10) NOT NULL,  
  Country String varchar(50) NOT NULL,  
  PhoneNumber varchar(14) NOT NULL,  
  
  DateOfBirth Date CHECK(date > todate('dd/mm/yy','01/01/1900')  
                           and date < now()),  
  
  Income numeric(5) CHECK(income >= 0 and income < 20000)  
);
```

Extras, exercises

Integridade, consistência

Em geral, exemplos, testes...

Integridade

- Uma base de dados está num estado de integridade se contém apenas dados válidos. Ou seja, se os dados armazenados na base de dados estão de acordo com a realidade.

Professor

🔑 BI	<u>Nome</u>	Universidade	Data_nascimento	Salario_mes
1234343	Joaquim Sousa	Univ. Coimbra	12-03-2060	- 900.00
1256673		Univ. Porto	18-01-1878	1215.56
1275432	Maria Oliveira	IST	15-09-1978	923.00
1022634	Joana Varela	IST	23-07-1988	623.218,01
1022634	Luísa Antunes	Univ. Coimbra	22-04-1989	647.00

Annotations:


- Não poder ser nulo: tem de ser preenchido (points to empty cell in Nome)
- Datas erradas (points to 12-03-2060)
- Não pode ser negativo (points to - 900.00)
- Demasiado grande (points to 623.218,01)
- Não pode haver duas chaves primárias iguais (points to duplicate BI values 1022634)

Integridade de domínio

- Regras de integridade que se aplicam a atributos de uma dada tabela.
- Os exemplos de violações de integridade na tabela Professor só podem acontecer se não forem definidas restrições de **integridade de domínio**.

UC

Professor

 BI	<u>Nome</u>	Universidade	Data_nascimento	Salario_mes
1234343	Joaquim Sousa	Univ. Coimbra	12-03-2060	- 900.00
1256673		Univ. Porto	18-01-1878	1215.56
1275432	Maria Oliveira	IST	15-09-1978	923.00
1022634	Joana Varela	IST	23-07-1988	623.218.01
1342664	Luísa Antunes	Univ. Coimbra	22-04-1989	647.00

Não poder ser nulo: tem de ser preenchido
 Datas erradas
 Não pode ser negativo
 Demasiado grande

bd

Aluno

A_NUM	NOME	End	TEL	BI
1002	A. Mendes	R. Sota, 23	12345	987654
1008	M. Melo	Av. Sá Band.	53428	457541

Disciplina

NOME	D_CODIG	ANO	SEM
Álgebra	ALGB	1	1
Electrónica I	ELEC	2	2

A_D

A_NUM	D_CODIG	EPOCA	NOTA
1008	ALGB	Set.	15

CREATE TABLE ALUNO

```

(A_NUM          NUMBER(4)          PRIMARY KEY,
 NOME           CHAR(60)          NOT NULL,
 END            CHAR(120),
 TEL           NUMBER(8),
 BI            NUMBER(4)          UNIQUE);

```

CREATE TABLE DISCIPLINA

```

(NOME           CHAR(60)          UNIQUE,
 D_CODIG        CHAR(6)          PRIMARY KEY,
 ANO            NUMBER(1)        CHECK ((ANO >= 1) AND (ANO <= 5)),
 SEM            NUMBER(1)        CHECK (SEM IN (1,2)));

```

CREATE TABLE A_D

```

(A_NUM          NUMBER(4),
 D_CODIG        CHAR(6),
 EPOCA          CHAR(6),
 NOTA           NUMBER(2)        CHECK ((NOTA >= 0) AND (NOTA <= 20)),
 CONSTRAINT A_D_CE1 FOREIGN KEY (A_NUM) REFERENCES ALUNO (A_NUM),
 CONSTRAINT A_D_CE2 FOREIGN KEY (D_CODIG) REFERENCES DISCIPLINA (D_CODIG),
 CONSTRAINT A_D_CP PRIMARY KEY (A_NUM, D_CODIG, EPOCA);

```

Violação de restrições de integridade: exemplos

Registos presentes nas tabelas

Aluno

A_NUM	NOME	End	TEL	BI
1002	A. Mendes	R. Sota, 23	12345	987654
1008	M. Melo	Av. Sá Band.	53428	457541

Disciplina

NOME	D_CODIG	ANO	SEM
Álgebra	ALGB	1	1
Electrónica I	ELEC	2	2

A_D

A_NUM	D_CODIG	EPOCA	NOTA
1008	ALGB	Set.	15

INSERT INTO A_D

VALUES (1004, 'ELEC', 'Julho', 12);

(Erro: viola a integridade referencial, chaves forasteira)

(—1004 — não tem correspondência na tabela de referência —Aluno—)

(Erro: viola UNIQUE em BI)

INSERT INTO ALUNO

VALUES (1010, 'A. Cavaco', 'R. de Baixo, 23', 2344, 987654)

INSERT INTO A_D

VALUES (1008, 'ALGB', 'Set'); (Correcto, apesar de NOTA fica Null)

INSERT INTO DISCIPLINA

VALUES ('Física', 'FISC', 0, 1); (Erro: viola integridade em Ano

```
CREATE TABLE ALUNO
(A_NUM      NUMBER(4)      PRIMARY KEY,
 NOME       CHAR(60)      NOT NULL,
 END        CHAR(120),
 TEL        NUMBER(8),
 BI         NUMBER(4)      UNIQUE);

CREATE TABLE DISCIPLINA
(NOME       CHAR(60)      UNIQUE,
 D_CODIG    CHAR(6)       PRIMARY KEY,
 ANO        NUMBER(1)     CHECK ((ANO >= 1) AND (ANO <= 5)),
 SEM        NUMBER(1)     CHECK (SEM IN (1,2)));

CREATE TABLE A_D
(A_NUM      NUMBER(4),
 D_CODIG    CHAR(6),
 EPOCA      CHAR(6),
 NOTA       NUMBER(2)     CHECK ((NOTA >= 0) AND (NOTA <= 20)),
 CONSTRAINT A_D_CE1 FOREIGN KEY (A_NUM) REFERENCES ALUNO (A_NUM),
 CONSTRAINT A_D_CE2 FOREIGN KEY (D_CODIG) REFERENCES DISCIPLINA (D_CODIG),
 CONSTRAINT A_D_CP PRIMARY KEY (A_NUM, D_CODIG, EPOCA);
```

Violação de restrições de integridade

Registos presentes nas tabelas

Aluno

A_NUM	NOME	End	TEL	BI
1002	A. Mendes	R. Sota, 23	12345	987654
1008	M. Melo	Av. Sá Band.	53428	457541

Disciplina

NOME	D CODIG	ANO	SEM
Álgebra	ALGB	1	1
Electrónica I	ELEC	2	2

A D

A_NUM	D CODIG	EPOCA	NOTA
1008	ALGB	Set.	15

UPDATE ALUNO

```
SET NOME = 'D. Mendes',
    A_NUM = 1006
WHERE A_NUM = 1008;
```

(Erro: viola a integridade referencial pois altera dado de referência correspondente à chave externa A_NUM da tabela ALUNO, ficando valor inexistente 1008 em A_D – tabela referenciada)

UPDATE A_D

```
SET D_CODIG = 'ELEC',
    NOTA = 14;
```

(Correcto, pois a chave externa D_CODIG é alterada mas mantem um valor (que existe na tabela Disciplina))

DELETE FROM ALUNO

```
WHERE A_NUM = 1008;
```

(Erro: viola a integridade referencial pois há dados dependentes do registo (a apagar na tabela A_D))

Integridade através de abordagem declarativa

Vantagens:

- Evita a necessidade de programas para garantir a integridade dos dados;
- Definição simples e rápida;
- Testes de integridade otimizados pelo SQL;
- Regras centralizadas e controladas pelo SGBD;
- É fácil alterar restrições sem mexer nas aplicações;
- Flexibilidade em activar/ desactivar restrições.

Desvantagens

- A validação dos dados não é imediata (só é feita na execução do INSERT ou UPDATE); => TRATA EXCEPÇÕES

Validação de integridade

1. Integridade definida declarativamente na BD (create table)
 - quando tento uma operação que viole integridade, dá erro
 - Numa aplicação cliente, isso gera uma exceção, que devo tratar

```
try {  
    // Your risky code goes between these curly braces!!!  
} catch(Exception ex) {  
    // Your exception handling code goes between these // curly braces,  
    // similar to the exception clause // in a PL/SQL block.  
} finally {  
    // Your must-always-be-executed code goes between these // curly  
    // braces. Like closing database connection.  
}
```

getErrorCode()	Gets the error number associated with the exception.
getMessage()	Gets the JDBC driver's error message for an error, handled by the driver or gets the Oracle error number and message for a database error.

Validação de inputs

1. Validação de campos no interface com utilizador

- Código (e.g. javascript) para validar campos inseridos
- Evita inserir dados proibidos
- Mas, ter em atenção duas coisas:
 - alguns aspectos não podem ser testados exclusivamente no IU (e.g. integridade de entidade e referencial)
 - quando possível é importante incluir também o mais possível integridade na própria BD, em vez de deixar ao critério dos programadores de aplicações clientes, que podem esquecer, não ter uniformidade (pode haver várias aplicações clientes), etc.

A BD tem de garantir consistência e integridade SEMPRE, não é opcional

Regras de negócio

- Há regras de integridade mais complexas que não podem ser definidas na estrutura das tabelas. Normalmente estas são verificadas através de funções escritas especificamente para o efeito que são executadas sempre que os dados são alterados (inseridos, actualizados ou apagados).
- Essas funções podem ser parte da aplicação cliente, ou escritas como funções a executar no proprio servidor (pl/sql)
- Não há problema nenhum em que sejam na aplicação cliente. O que se torna essencial é **UMA ADEQUADA DOCUMENTAÇÃO DESTES DETALHES NO DESENVOLVIMENTO DO MODELO DE DADOS -> normalmente, isto falha!**
- Exemplos:
 - Um empregado não pode ganhar mais do que o seu chefe.
 - O salário de um empregado não pode ser reduzido (só pode ser aumentado).
 - Não se pode inserir registos numa dada tabela entre as 18 horas e as 8 horas da manhã.
 - Etc.