Structured Query Language - SQL

- Tópicos:
  - Linguagem de definição de dados
  - Estrutura básica de perguntas em SQL
  - Operações com conjuntos
  - Funções de agregação
  - Valores nulos
  - Junções
  - Subconsultas
  - Vistas e relações derivadas
  - Modificações de bases de dados
  - Ligação com outras linguagens de programação
- Bibliografia:
  - Capítulo 3 e Secções 4.1, 4.2, 5.1 e 5.2 do livro recomendado

A linguagem SQL

- Até agora vimos as bases (teóricas) de modelação e manipulação de bases de dados
  - Mas como é que tudo isto se usa na prática?
- A linguagem SQL (Stuctured Query Language), é um standard usado por (praticamente) todos os Sistemas de Gestão de Bases de Dados relacionais.
- Incluí:
  - **Linguagem de definição de dados** (DDL), que permite "criar" bases de dados relacionais de acordo com o modelo relacional
  - **Linguagem de manipulação de dados** (DML), que permite interrogar e manipular bases de dados de forma declarativa, e que tem as suas bases na álgebra relacional

Data Definition Language (DDL)

- Linguagem para especificar informação das relações, incluindo:
  - O esquema de cada relação.
  - O domínio de valores associados com cada atributo.
  - Restrições de integridade (incluindo chaves, chaves estrangeiras)

- Também permite dar indicações sobre
  - Estruturas de armazenamento físico em disco de cada relação
  - Estruturas de dados para tornar o acesso mais eficiente
  - Informação de segurança e autorização para cada relação.

Instrução Create Table

- Uma tabela SQL é definida recorrendo ao comando **create table**

  create table r (A_1 D_1, A_2 D_2, ..., A_n D_n,
  (integrity-constraint$_1$),
  ...,
  (integrity-constraint$_k$));

  - r é o nome da relação
  - A_i é o nome de um atributo no esquema de relação r
  - D_i é o tipo de dados dos valores no domínio do atributo A_i

- Exemplo:

  create table *branch*
  (*branch_name* char(15) **not null,**
  branch_city char(30),
  assets integer);

Tipos em SQL

- **char(n).** Cadeia de caracteres de comprimento fixo *n.*
- **varchar(n).** Cadeia de caracteres de comprimento variável, com o máximo *n* caracteres, especificado pelo utilizador.
- **int.** inteiro (um subconjunto finito dos inteiros, dependente da máquina).
- **smallint.** Inteiro pequeno (um subconjunto do tipo int).
- **numeric(p,d).** Número de vírgula fixa, com precisão de *p* dígitos e com *d* casas decimais.
- **real, double precision.** Números de vírgula flutuante, com precisão dependente da máquina.
- **float(n).** Número de vírgula flutuante, com um mínimo de precisão de *n* dígitos.
- Os valores nulos são permitidos em todos os tipos de dados. A declaração de um atributo como **not null** proíbe os valores nulos para esse atributo.
- **create domain** construção em SQL-92 que cria um tipo de dados definido pelo utilizador (não se encontra implementado em Oracle11g)

  create domain *person-name* **char**(20) **not null**

Tipos em Oracle

- Alguns tipos suportados pelo SGBD Oracle11g são:
  - CHAR(size), VARCHAR2(size), NCHAR(size), NVARCHAR2(size)
  - NUMBER(precision,scale)
  - DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE e TIMESTAMP WITH LOCAL TIME ZONE
  - INTERVAL YEAR TO MONTH e INTERVAL DAY TO SECOND.
  - RAW(size), LONG RAW, LONG
  - CLOB, NCLOB, BLOB, BFILE
  - ROWID, UROWID

Conversão (implícita) para tipos SQL

CHARACTER(n) CHAR(n)	CHAR(n)
CHARACTER VARYING(n) CHAR VARYING(n)	VARCHAR2(n)
NATIONAL CHARACTER(n) NATIONAL CHAR(n) NCHAR(n)	NCHAR(n)
NATIONAL CHARACTER VARYING(n) NATIONAL CHAR VARYING(n) NCHAR VARYING(n)	NVARCHAR2(n)
INTEGER INT SMALLINT	NUMBER(38)
FLOAT(b) DOUBLE PRECISION REAL	NUMBER

Tipos Data/Tempo em SQL (cont.)

- **date.** datas, contendo um ano com (4 dígitos), mês e dia
  - E.g. **date** '2001-7-27'
- **time.** Tempo (diário), em horas, minutos e segundos.
  - E.g. **time** '09:00:30' **time** '09:00:30.75'
- **timestamp**: data mais hora
  - E.g. **timestamp** '2001-7-27 09:00:30.75'
- **Interval**: período de tempo
  - E.g. Interval '1' day
  - A subtracção de dois valores de date/time/timestamp devolve um intervalo
  - Os valores de intervalos podem ser adicionados a valores de date/time/timestamp
- Pode-se extrair campos do valor date/time/timestamp
  - E.g. **extract** (**year from** SYSDATE)
- Pode-se converter cadeias de caracteres para date/time/timestamp
  - E.g. **cast** <string-valued-expression> **as date**

Datas/Tempo em Oracle

- O suporte de datas em SGBD Oracle11g difere ligeiramente do standard
- Os tipos base para representar datas/tempos são:
  - DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE e TIMESTAMP WITH LOCAL TIME ZONE.
- Para representar intervalos temos:
  - INTERVAL YEAR TO MONTH e INTERVAL DAY TO SECOND.
- Existem funções especiais para converter de e para datas/intervalos:
  - TO_CHAR(date), TO_NCHAR(date).
  - TO_DATE, TO_TIMESTAMP, TO_TIMESTAMP_TZ, TO_YMINTERVAL, TO_DSINTERVAL, NUMTOYMINTERVAL, NUMTODSINTERVAL
- A pseudocoluna SYSDATE e função CURRENT_DATE retornam a data atual.
- Para mais detalhes consultar o manual.

Restrições de integridade em Create Table

- **not null**
- **primary key** (A_1, …, A_n)
- **unique** (A_1, …, A_n)
- **check** *(P),* em que *P* é um predicado
- **foreign key** (A_1, …, A_n) **references** R(B_1, …, B_n)

Exemplo:

create table *account*
(account_number integer**,**
branch_name char(30) **not null**,
balance integer **not null**,
primary key *(account-number),*
foreign key *(branch-name)* **references** *branch(branch_name)*,
check *(balance* >= 0));

A declaração **primary key** num atributo garante automaticamente **not null**

Instrução Drop Table

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

drop table loan

- Em Oracle, caso existam restrições de integridade (e.g. chaves estrangeiras) associadas à tabela, deve-se utilizar a declaração **cascade constraints**.

drop table loan cascade constraints

- Para eliminar vistas (veremos mais à frente o que são) deve-se utilizar **drop view**

Instrução Alter Table

- O comando **alter table** é utilizado para modificar o esquema, ou as restrições sobre relações já existente.
- Para adicionar novos atributos:

  alter table *r* **add** *A D*

  em que *A* é o nome do atributo a adicionar à relação *r* e *D* o domínio de *A.* Todos os tuplos existentes ficam com *null* no novo atributo.
- O comando **alter table** também pode ser utilizado para eliminar atributos de uma relação

  alter table *r* **drop** *A*

  em que *A* é o nome de um atributo na relação *r* (em Oracle deve-se escrever **alter table** *r* **drop column** *A)*

Instrução Alter Table

- Para adicionar novas restrições:

  alter table *r* **add constraint** *N R*

  em que *N* é um nome dado à nova restrição e R define a restrição. Por exemplo:

  alter table *account* **add constraint** *saldo_pos*
  check (*balance > 0***)**

- Para remover restrições (definidas previamente com um nome):

  alter table *r* **drop constraint** *N*

  por exemplo:

  alter table *account* **drop constraint** *saldo_pos*

Exemplo retirado das aulas práticas

```
create table alunos(
        num_aluno number(6) not null,
        nome varchar2(30) not null,
        local varchar2(25),
        data_nsc date not null,
        sexo char(1) not null check ( sexo in ( 'F' , 'M' )  ),
        cod_curso number(3) not null);
…
alter table alunos add constraint pk_aluno
                        primary key (num_aluno);
alter table alunos add constraint un_aluno
                        unique (num_aluno, cod_curso);
…
```

Estrutura básica de pergunta na Data Manipulation Language (DML)

- SQL é baseada em operações de conjuntos e de álgebra relacional com algumas modificações e extensões
- Uma consulta SQL básica tem a forma:

  select A_1, A_2, ..., A_n
  from r_1, r_2, ..., r_m
  where P

  - A_i*s* representam atributos
  - r_i*s* representam relações
  - P é um predicado.
- A consulta é equivalente à expressão de álgebra relacional (a menos de tratamento de repetições):

$$\prod_{A1,\ A2,\ ...,\ An}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m))$$

- O resultado de uma consulta SQL é uma relação (possivelmente com repetições).

A cláusula select

- A cláusula **select** corresponde à operação de projeção da álgebra relacional. É utilizada para listar os atributos pretendidos no resultado da consulta.
- Listar os nomes de todas as agências na relação *loan*

  select *branch_name*
  from *loan*
- Na sintaxe de álgebra relacional a consulta seria:

  $\Pi_{\text{branch-name}}(loan)$
- Um asterisco na cláusula select denota "todos os atributos" (elimina a projeção)

  select *
  from *loan*
- NOTA: O SQL não permite o caracter '-' nos nomes, portanto deverá utilizar, por exemplo, *branch_name* em vez de *branch-name* num sistema existente.
- NOTA: As maiúsculas e minúsculas não são distinguidas em nomes da linguagem SQL.
  - Poderá utilizar maiúsculas nos sítios onde utilizamos **bold**.

A cláusula select (cont.)

- O SQL permite duplicados nas relações e nos resultados de consultas.
- Para forçar a eliminação de duplicados, inserir a palavra-chave **distinct** após **select.**
  - Apresentar os nomes de todos os balcões onde foram efetuados empréstimos, sem repetições

    select distinct *branch_name*
    from *loan*

- A palavra-chave **all** indica que os duplicados não devem ser removidos (por default, o **all** é assumido pelo sistema, mesmo que não o coloque lá)

    select all *branch_name*
    from *loan*

A cláusula select (cont.)

- A cláusula **select** pode conter expressões aritméticas envolvendo as operações, +, –, ∗, e /, com argumentos constantes ou atributos de tuplos (como na projeção generalizada). Dependendo das implementações, encontra-se normalmente definida uma biblioteca de funções.

- A consulta:

  select *loan_number, branch_name, amount* ∗ 100
  from *loan*

  devolve uma relação idêntica à relação *loan*, exceto que o atributo *amount* é multiplicado por 100 (como na projeção generalizada).

A cláusula where

- A cláusula **where** corresponde ao predicado de seleção da álgebra relacional. É formada por um predicado envolvendo atributos de relações que aparecem na cláusula **from**.
  - Para encontrar os números de contas da agência da Caparica com saldos superiores a 100.

    select *account_number*
    from *account*
    where *branch_name* = 'Caparica' **and** *balance* > 100
- Os resultados de comparações podem ser combinados por intermédio dos conectivos lógicos **and, or,** e **not.**
- Podem-se aplicar comparações ao resultado de expressões aritméticas.

A cláusula where (cont.)

- A linguagem SQL possui um operador de comparação **between** para especificar condições em que um valor deve estar contido num intervalo de valores (incluindo os seus extremos)
  - Apresentar os números dos empréstimos de montantes entre €90,000 e €100,000 (ou seja, ≥€90,000 e ≤€100,000)

```
select loan_number
from loan
where amount between 90000 and 100000
```

- Para negar a condição pode-se colocar o conetivo **not** antes de **between**

Operações com Cadeias de Caracteres

- O SQL inclui um mecanismo de concordância de padrões para comparações envolvendo cadeias de caracteres. Os padrões são descritos recorrendo a dois caracteres especiais:
  - percentagem(%). O carácter % concorda com qualquer subcadeia.
  - sublinhado (_). O carácter _ concorda com qualquer caracter.
- Listar todos os clientes cuja rua inclua a subcadeia “Main”.

```
select customer_name
from customer
where customer_street like ‘%Main%’
```

- Concordar com o nome “Main%”

```
like ‘Main\%’ {escape  ‘\’}
```

- A SQL suporta uma variedade de operações com cadeias de caracteres, tais como:
  - concatenação (utilizando “||”)
  - conversão para maiúsculas (**upper**) e para minúsculas (**lower**)
  - calcular o comprimento, extração de subcadeias, etc.

A cláusula from

- A cláusula **from** corresponde à operação de produto cartesiano da álgebra relacional. Indica as relações a consultar na avaliação da expressão.
  - Encontrar o produto cartesiano *borrower x loan*

    select *
    from *borrower, loan*
  - Listar o nome, número de empréstimo e montante de todos os clientes que efetuaram um empréstimo na agência de Perryridge.

    select *borrower.*, amount*
    from *borrower, loan*
    where *borrower.loan_number = loan.loan_number* **and**
    branch_name = 'Perryridge'

A operação de Renomeação

- A linguagem SQL permite a renomeação de relações e atributos recorrendo à cláusula **as** :

  old_name **as** *new_name*

- Listar o nome, número de empréstimo e montante de todos os clientes, renomeando o nome da coluna *loan_number* para *loan_id.*

  select *customer_name, borrower.loan_number* **as** *loan_id, amount*
  from *borrower, loan*
  where *borrower.loan_number = loan.loan_number*

- Caso se pretenda utilizar um nome com espaços, esse nome deverá ser colocado entre **aspas**.

Variáveis de tuplo

- As variáveis de tuplo são definidas na cláusula **from** por intermédio da cláusula **as** opcional (No Oracle não deverá colocar o **as**).
  - Apresente os nomes de todos os clientes e respetivos números dos empréstimos que possuam um empréstimo nalguma agência.

    select *customer_name, T.loan_number, S.amount*
    from *borrower* **as** *T, loan* **as** *S*
    where *T.loan_number = S.loan_number*

  - Liste todas as agências que têm mais ativos do que pelo menos uma agência localizada em Brooklyn

    select distinct *T.branch_name*
    from *branch* **as** *T, branch* **as** *S*
    where *T.assets > S.assets* **and** *S.branch_city* = 'Brooklyn'

  - Em Oracle...

    select distinct *T.branch_name*
    from *branch T, branch S*
    where *T.assets > S.assets* **and** *S.branch_city* = 'Brooklyn'

Variáveis de tuplo (cont)

- As variáveis de tuplos podem ser vistas como criando várias cópias de uma mesma relação (similar às renomeações da A.R.)
- **Exemplo:** Considere a relação:

  voos(numVoo,Matr,Data,Hora,De,Para)

  em que cada tuplo denota um voo com nº *numVoo*, efectuado do aeroporto *De* para o aeroporto *Para* no dia *Data* à hora *Hora* no avião com Matrícula *Matr.*
- Quais os pares de voos que usaram o mesmo avião num mesmo dia?

```
select distinct T.numVoo, S.numVoo
from voos as T, voos as S
where T.Matr = S.Matr
   and T.Data = S.Data
   and T.numVoo < S.numVoo
```

Select Case

- O SGBD Oracle11g possui uma extensão muito útil que permite a utilização de condições **if-then-else**:

```
select numero,
        case  when sexo='M' then 'Senhor ' || nome
              when sexo='F' then 'Senhora ' || nome
        end as titNome
from alunos
```

- Quando a expressão a comparar é a mesma tem-se a forma

```
select numero,
        case  sexo when 'M' then 'Senhor ' || nome
                   when 'F' then 'Senhora ' || nome
        end as titNome
from alunos
```

Select Case

- Pode-se ter um ramo **else** final:

```
select alu_numero as aluno, cad_codigo as cadeira,
    case    when nota='E' then 'Excluído'
            when nota='A' then 'Ausente  '
            when nota='F' then 'Faltou'
            when nota is null then 'Nota por lançar'
            else nota
    end
from inscricoes
```

- O CASE pode ocorrer em qualquer lugar onde se pode ter uma expressão (por exemplo na cláusula **select**, **where** ou mesmo dentro de funções de agregação – veremos adiante o que é isto)

Biblioteca Funções Oracle

- Funções com números:
  - ABS, ACOS, ASIN, ATAN, ATAN2, BITAND, CEIL, COS, COSH, EXP, FLOOR, LN, LOG, MOD, POWER, ROUND, SIGN, SIN, SINH, SQRT, TAN, TANH, TRUNC, WIDTH_BUCKET
- Funções com cadeias de caracteres retornando cadeias:
  - CHR, CONCAT, INITCAP, LOWER, LPAD, LTRIM, NLS_INITCAP, NLS_LOWER, NLSSORT, NLS_UPPER, REPLACE, RPAD, RTRIM, SOUNDEX, SUBSTR, TRANSLATE, TREAT, TRIM, UPPER
- Funções com cadeias de caracteres retornando números
  - ASCII, INSTR, LENGTH
- Funções com datas e intervalos de tempos
  - ADD_MONTHS, CURRENT_DATE, CURRENT_TIMESTAMP, DBTIMEZONE, EXTRACT (datetime), FROM_TZ, LAST_DAY, LOCALTIMESTAMP, MONTHS_BETWEEN, NEW_TIME, NEXT_DAY, NUMTODSINTERVAL, NUMTOYMINTERVAL, ROUND, SESSIONTIMEZONE, SYS_EXTRACT_UTC, SYSTIMESTAMP, SYSDATE, TO_DSINTERVAL, TO_TIMESTAMP, TO_TIMESTAMP_TZ, TO_YMINTERVAL, TRUNC (date), TZ_OFFSET

Biblioteca Funções Oracle

- Funções de Conversão
  - ASCIISTR, BIN_TO_NUM, CAST, CHARTOROWID, COMPOSE, CONVERT, DECOMPOSE, HEXTORAW, NUMTODSINTERVAL, NUMTOYMINTERVAL, RAWTOHEX, RAWTONHEX, ROWIDTOCHAR, ROWIDTONCHAR,TO_CHAR (character), TO_CHAR (datetime), TO_CHAR (number), TO_CLOB, TO_DATE, TO_DSINTERVAL, TO_LOB, TO_MULTI_BYTE, TO_NCHAR (character), TO_NCHAR (datetime), TO_NCHAR (number), TO_NCLOB, TO_NUMBER, TO_SINGLE_BYTE, TO_YMINTERVAL, TRANSLATE ... USING, UNISTR
- Outras
  - BFILENAME, COALESCE, DECODE, DUMP, EMPTY_BLOB, EMPTY_CLOB, EXISTSNODE, EXTRACT (XML), GREATEST, LEAST, NLS_CHARSET_DECL_LEN, NLS_CHARSET_ID, NLS_CHARSET_NAME, NULLIF, NVL, NVL2, SYS_CONNECT_BY_PATH, SYS_CONTEXT, SYS_DBURIGEN, SYS_EXTRACT_UTC, SYS_GUID, SYS_TYPEID, SYS_XMLAGG, SYS_XMLGEN, UID, USER, USERENV, VSIZE

Ordenação de tuplos

- Listar em ordem alfabética os nomes de todos os clientes que possuem um empréstimo na agência de Perryridge

```
select distinct customer_name
from    borrower, loan
where borrower.loan_number = loan.loan_number and
        branch_name = 'Perryridge'
order by customer_name
```

- Pode-se especificar **desc** para ordenação descente ou **asc** para ordenação ascendente, para cada atributo; por omissão, assume-se ordem ascendente.
  - E.g. **order by** *customer_name* **desc**
- Pode-se especificar **nulls first** ou **nulls last**, após o **desc/asc**, para indicar como ordenar os valores nulos (SQL:2003).
  - E.g. **order by** assets **desc nulls last**
- Pode-se ter mais do que uma chave de ordenação, separando-as com vírgulas

Estrutura Básica

- SQL é baseada em operações de conjuntos e de álgebra relacional com algumas modificações e extensões
- Uma consulta SQL básica tem a forma:

  select A_1, A_2, ..., A_n
  from r_1, r_2, ..., r_m
  where P

  - A_i*s* representam atributos
  - r_i*s* representam relações
  - P é um predicado.
- A consulta é equivalente à expressão de álgebra relacional (a menos de tratamento de repetições):

$$\prod_{A1,\, A2,\, ...,\, An}(\sigma_P (r_1 \times r_2 \times \ldots \times r_m))$$

- O resultado de uma consulta SQL é uma relação (possivelmente com repetições).

Duplicados

- Em relações com duplicados, a linguagem SQL especifica quantas cópias dos tuplos aparecem no resultado.
- *Versões Multiconjunto* de alguns operadores da álgebra relacional dadas relações multiconjunto r_1 e r_2:
  1. Se existem c_1 cópias do tuplo t_1 em r_1, e t_1 satisfaz a seleção σ_p, então existem c_1 cópias de t_1 em $\sigma_p(r_1)$.
  2. Para cada cópia do tuplo t_1 em r_1, existe uma cópia do tuplo $\Pi_A(t_1)$ em $\Pi_A(r_1)$, onde $\Pi_A(t_1)$ denota a projeção do tuplo t_1.
  3. Se existem c_1 cópias do tuplo t_1 em r_1 e c_2 cópias do tuplo t_2 em r_2, então existem c_1 x c_2 cópias do tuplo t_1. t_2 em r_1 x r_2

Duplicados (cont.)

- Exemplo: supondo que as relações multiconjunto r_1 (*A, B*) e r_2 (*C*) são as seguintes:

$$r_1 = \{(1, a)\ (2, a)\} \qquad r_2 = \{(2), (3), (3)\}$$

- Então $\Pi_B(r_1)$ devolve {(a), (a)}, enquanto que $\Pi_B(r_1)$ x r_2 é

$$\{(a,2), (a,2), (a,3), (a,3), (a,3), (a,3)\}$$

- A semântica com duplicados da consulta SQL:

select A_1,, A_2, ..., A_n
from r_1, r_2, ..., r_m
where *P*

é equivalente à versão *multiconjunto* da expressão:

$$\Pi_{A1,, A2, ..., An}(\sigma_P (r_1 \text{ x } r_2 \text{ x } ... \text{ x } r_m))$$

Operações com Conjuntos

- As operações com conjuntos **union, intersect,** e **except (minus** no Oracle11g) operam sobre relações e correspondem aos operadores de álgebra relacional $\cup$, $\cap$ e -
- Ao contrário das outras operações em SQL, as operações com conjuntos eliminam os duplicados automaticamente. Para reter duplicados deve-se utilizar as respectivas versões multiconjunto **union all, intersect all** e **except all.**

  Supondo que um tuplo ocorre *m* vezes em *r* e *n* vezes em *s,* então ele ocorre:

  - *m* + *n* vezes em *r* **union all** *s*
  - min(*m,n)* vezes em *r* **intersect all** *s*
  - max(0, *m* – *n)* vezes em *r* **except all** *s*

Operações com Conjuntos

- Listar todos os clientes que têm um empréstimo ou uma conta:

  (select *customer_name* **from** *depositor*)
  union
  (select *customer_name* **from** *borrower)*

- Listar todos os clientes que têm um empréstimo e uma conta:.

  (**select** *customer_name* **from** *depositor*)
  intersect
  (select *customer_name* **from** *borrower)*

- Listar os clientes que têm uma conta mas não têm empréstimos

  (**select** *customer_name* **from** *depositor*)
  except
  (select *customer_name* **from** *borrower)*