ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Structured Query Language

SubQueries

DEE
Departamento de Engenharia Electrotécnica

Licenciatura em Engenharia de Telecomunicações e Informática

MANIPULANDO DADOS DE DATA E TEMPO

FUNÇÃO	PARAMETROS
DATEADD	(datepart, number, date)
DATEDIFF	(datepart, date1, date2)
DATENAME	(datepart, date)
DATEPART	(datepart, date)
GETDATE	()

- *Datepart* - É o parâmetro que especifica à qual parte da data deve-se adicionar um número (se usamos o DATEADD).
- *Number* - É o valor usado para incrementar *datepart*. O valor deve ser um valor inteiro conhecido quando a expressão é analisada.
- *Date* - É uma expressão que retorna uma data válida ou uma cadeia de caracteres em formato de data.

Funções Data e Tempo - Exemplos

```
SELECT DAY('2017/08/25') AS DayOfMonth;
```

DayOfMonth
25

```
SELECT MONTH('2017/08/25') AS Month;
```

Month
8

```
SELECT YEAR('2017/08/25') AS Year;
```

Year
2017

```
SELECT GETDATE();
```

2021-11-21 16:34:31.250

```
SELECT DATEADD(year, 1, '2017/08/25') AS DateAdd;
```

```
DATEADD(interval, number, date)
```

DateAdd
2018-08-25 00:00:00.000

SET LANGUAGE English
SELECT DATENAME(DW, GETDATE()) AS 'Nome do dia em English'

Funções Data e Tempo - Exemplos

```
SELECT DATEDIFF(year, '2017/08/25', '2011/08/25') AS DateDiff;
```

```
DATEDIFF(interval, date1, date2)
```

DateDiff
-6

```
SELECT DATEPART(year, '2017/08/25') AS DatePartInt;
```

```
DATEPART(interval, date)
```

DatePartInt
2017

SELECT TOP clause

- A cláusula SELECT TOP permite limitar o número de linhas ou a percentagem de linhas retornadas em um conjunto de resultados de consulta.
- A instrução SELECT TOP é sempre usada em conjunto com a cláusula ORDER BY. Portanto, o conjunto de resultados é limitado ao primeiro número N de linhas ordenadas.

```
SELECT TOP (expression) [PERCENT]
    [WITH TIES]
FROM
    table_name
ORDER BY
    column_name;
```

```
SELECT TOP 10
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price DESC;
```

Este exemplo usa um valor constante para retornar os 10 produtos mais caros.

PIVOT CLAUSE

- permite escrever uma tabulação cruzada.
  - Isso significa que se pode agregar os resultados de consulta e girar as linhas em colunas.

```
SELECT first_column AS <first_column_alias>,
[pivot_value1], [pivot_value2], ... [pivot_value_n]
FROM
(<source_table>) AS <source_table_alias>
PIVOT
(
 aggregate_function(<aggregate_column>)
 FOR <pivot_column>
 IN ([pivot_value1], [pivot_value2], ... [pivot_value_n])
) AS <pivot_table_alias>;
```

employee_number	last_name	first_name	salary	dept_id
12009	Sutherland	Barbara	54000	45
34974	Yates	Fred	80000	45
34987	Erickson	Neil	42000	45
45001	Parker	Sally	57500	30
75623	Gates	Steve	65000	30

TotalSalaryByDept	30	45
TotalSalary	122500	176000

```
SELECT 'TotalSalary' AS TotalSalaryByDept,
[30], [45]
FROM
(SELECT dept_id, salary
 FROM employees) AS SourceTable
PIVOT
(
 SUM(salary)
 FOR dept_id IN ([30], [45])
) AS PivotTable;
```

EXEMPLOS

Inscritos

Id_aluno	Id_disciplina
1090	BASDAD
1090	LPROJ
1080	BASDAD
1070	BASDAD
1060	BASDAD
1060	LPROJ

Disciplina

Id_disciplina	descricao
BASDAD	Base de Dados
LPROJ	Laboratorio projeto

Pergunta: Quais são os alunos inscritos em **todas** as disciplinas?

Id_aluno	Id_disciplina
1090	BASDAD
1090	LPROJ
1060	BASDAD
1060	LPROJ

CONSULTA - TODOS

Pergunta: Quais são os alunos inscritos em **todas** as disciplinas?

- **por Inclusão de Conjuntos - usa-se EXISTS / IN e EXCEPT**

```
SELECT I1.ID_ALUNO
FROM INSCRITOS I1
WHERE NOT EXISTS ( SELECT ID_DISCIPLINA --------- todas as disciplinas
                   FROM DISCIPLINA
                   EXCEPT
                   SELECT ID_DISCIPLINA --------- todas as disciplinas de I1
                   FROM INSCRITOS I2
                   WHERE I2.ID_ALUNO = I1.ID_ALUNO);
```

Se o aluno 1090 **(i1.id_aluno)** estiver inscrito em todas as disciplinas,

então: DISCIPLINAS – $\sigma_{ID\_ALUNO=1090}(INSCRITOS) = \emptyset$

CONSULTA - TODOS

Pergunta: Quais são os alunos inscritos em **todas** as disciplinas?

- **por Comparação de Cardinalidades – usa-se GROUP BY e COUNT()**

```
SELECT ID_ALUNO
FROM INSCRITOS
GROUP BY ID_ALUNO
HAVING COUNT(*)=( SELECT COUNT(*)
                  FROM DISCIPLINA)
```

-nr. disciplinas aluno = total disciplinas

Id_disciplina	descricao
BASDAD	Base de Dados
LPROJ	Laboratorio projeto

Id_aluno	Id_disciplina
1090	BASDAD
1090	LPROJ
1080	BASDAD
1070	BASDAD
1060	BASDAD
1060	LPROJ

CONSULTA - TODOS

Pergunta: Quais são os alunos inscritos em **todas** as disciplinas?

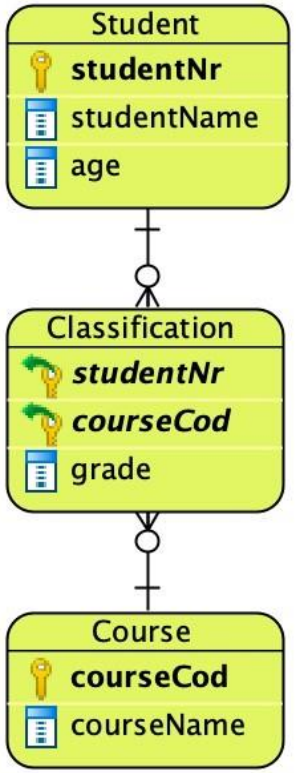
- **por Quantificação - usa-se EXISTS e/ou IN**
  - Reformulando a pergunta:
    - "Quais são os alunos, para os quais não existe nenhuma disciplina sem a sua inscrição?"

```
SELECT DISTINCT ID_ALUNO
FROM INSCRITOS I1
WHERE NOT EXISTS (
SELECT *
                FROM DISCIPLINA D WHERE NOT EXISTS (
    SELECT *
    FROM INSCRITOS I2
    WHERE I2.ID_ALUNO = I1.ID_ALUNO
    AND I2.ID_DISCIPLINA = D.ID_DISCIPLINA));
```

SUBQUERIES

Consultas numa BDR

Dada a UoD/domínio "notas de cursos" representada pelo seguinte modelo de dados relacionais:



- Instância BDR:

	STUDENTNR	STUDENTNAME	AGE
1	1	Ana	30
2	2	João	35
3	3	Maria	40
4	4	Rui	30
5	5	Joana	35
6	6	José	35
7	7	Rita	40
8	8	Camilo	33

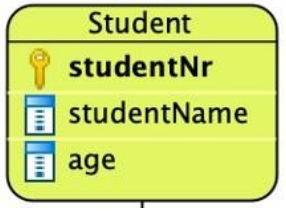
	STUDENTNR	COURSECOD	GRADE
1	1	info	15
2	2	info	18
3	3	info	18
4	4	info	14
5	5	info	12
6	6	info	18
7	7	info	14
8	8	info	14
9	1	math	13
10	2	math	16
11	3	math	19
12	6	math	17
13	7	math	15
14	8	math	10
15	1	hist	16
16	2	hist	14
17	3	hist	17
18	4	hist	12
19	5	hist	14
20	7	hist	12
21	8	hist	15
22	1	engl	18
23	2	engl	16
24	3	engl	19
25	4	engl	13
26	5	engl	13
27	8	engl	14

	COURSECOD	COURSENAME
1	info	Informatics
2	math	Math
3	hist	History
4	engl	English

SUBQUERIES

1. Mostre o número, nome e idade dos alunos mais novos, utilizando duas estratégias diferentes:
   - Com função de agregação MIN
   - Com o operador ALL

```
--1.a)
SELECT studentNr, studentName, age
  FROM student
 WHERE age = (SELECT MIN(age) FROM student);
```



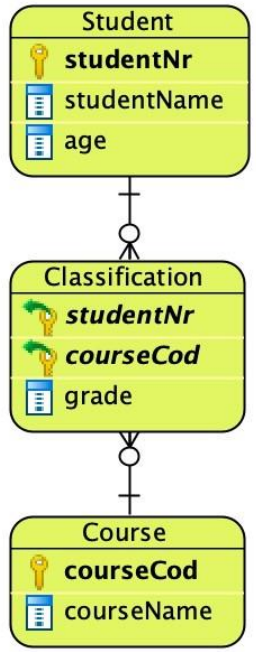

```
--1.b)
SELECT studentNr, studentName, age
  FROM student
 WHERE age <= ALL(SELECT age FROM student);
```

	NRALUNO	NOMEALUNO	IDADE
1	1	Ana	30
2	4	Rui	30

SUBQUERIES

2. Mostre o número e nome dos alunos que não têm nota no curso de matemática.

```
SELECT a.studentNr, a.studentName
  FROM student a
 WHERE a.studentNr NOT IN (    SELECT c.studentNr
                                 FROM classification c
                           INNER JOIN course d ON d.courseCod=c.courseCod
                                WHERE UPPER(d.courseName) LIKE 'MATH');
```



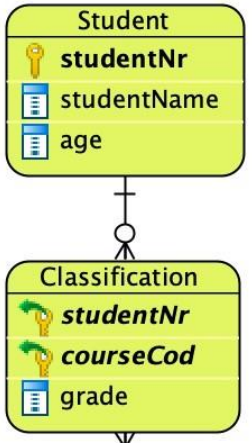
	STUDENTNR	STUDENTNAME
1	4	Rui
2	5	Joana

```
SELECT a.studentNr, a.studentName
  FROM student a
 WHERE a.studentNr NOT IN ( SELECT c.studentNr
                              FROM classification c
                             WHERE c.courseCod = ( SELECT d.courseCod
                                                     FROM course d
                                                    WHERE UPPER(d.courseName) LIKE 'MATH'));
```

SUBQUERIES

3. Apresente a nota mais baixa de cada aluno (número e nome), por ordem decrescente.

```
SELECT a.studentNr,
       a.studentName,
       (SELECT MIN(grade)
          FROM classification c
         WHERE c.studentNr=a.studentNr) "min grade"
  FROM student a
ORDER BY 3 DESC;
```



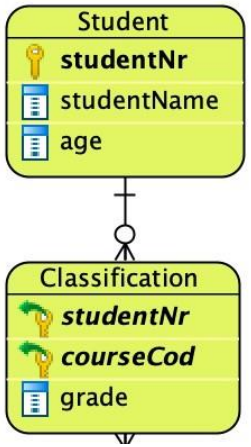

```
  SELECT a.studentNr, a.studentName, MIN(c.grade) "min grade"
    FROM student a
INNER JOIN classification c ON c.studentNr=a.studentNr
  GROUP BY a.studentNr, a.studentName
  ORDER BY 3 DESC;
```

	STUDENTNR	STUDENTNAME	min grade
1	6	José	17
2	3	Maria	17
3	2	João	14
4	1	Ana	13
5	4	Rui	12
6	5	Joana	12
7	7	Rita	12
8	8	Camilo	10

SUBQUERIES

4. Mostre o número e nome dos alunos que têm todas as notas superiores a 15.

```
SELECT a.studentNr, a.studentName
  FROM student a
 WHERE 15 <= ALL ( SELECT grade
                     FROM classification c
                    WHERE c.studentNr=a.studentNr);
```



	STUDENTNR	STUDENTNAME
1	3	Maria
2	6	José

Resumo da primeira parte da UC

- A saber:
  - Modelo Conceptual
  - Conceito de Normalização;
  - Modelo Lógico
  - Modelo Físico
  - Linguagem SQL
    - DDL
    - DML