

Introdução a Banco de Dados I

Caros alunos, as videoaulas desta disciplina encontram-se no AVA
(Ambiente Virtual de Aprendizagem).

Unidade 04

Padrão SQL e Estrutura de Sistemas de Arquivos

Introdução da Unidade

Esta unidade está direcionada a apresentar ao aluno sobre os principais comandos da linguagem SQL (*Select, insert, update, delete*) e também realizar uma abordagem em torno das estruturas de sistemas e arquivos (Estrutura geral, meios de armazenamento e organização de registros).

Objetivos

- Apresentar a introdução relacionada ao padrão SQL e os principais comandos utilizados;
- Apresentar abordagem para os comandos SQL (*Select, Insert, Update e Delete*);
- Abordar de forma geral sobre a estrutura de sistemas e arquivos;
- Abordar sobre meios de armazenamento e organização de registros.

Conteúdo programático

Aula 01 – Padrão SQL (*Select, Insert, Update e Delete*).

Aula 02 – Detalhamento sobre a estrutura geral, meios de armazenamento, organização de registros.



Você poderá, também, **assistir as videoaulas** em seu celular, basta apontar a câmera para os **QR Codes** distribuídos neste conteúdo.

Referências

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. 4. ed. São Paulo: Pearson Addison Wesley, 2005.

FRANÇA, Cícero Tadeu Pereira Lima; CELESTINO JÚNIOR, Joaquim. **Banco de dados**. 2. ed. Fortaleza: EdUECE, 2015.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistemas de banco de dados**. Rio de Janeiro: Elsevier, 2006. 781 p.

Aula 01 - Padrão SQL (*Select, Insert, Update e Delete*)

Padrão SQL

A *Structured Query Language* (Linguagem de Consulta Estruturada) ou mais conhecido publicamente como SQL, é uma linguagem padrão que permite padronizar a construção e acesso

ao Sistema de Gerenciamento de Banco de Dados (SGBD) de diferentes plataformas, indiferentemente de *softwares* ou de *hardwares*.

França (2015) aborda que o padrão SQL impulsiona não apenas a disseminação dos SGBDR, mas também a própria SQL. Para entender a importância da SQL, são mostrados nesta subseção todos os pontos que levaram os DBAs (*Database Administrator* – Administrador de Banco de Dados) a ter no SQL um aliado importante.

Conforme França (2015), os comandos SQL são divididos em dois subgrupos:

- **DDL – Data Definition Language:** subconjunto utilizado para criar, alterar e excluir tabelas e elementos associados; esse é o grupo que mais muda de um fabricante para outro;
- **DML – Data Manipulation Language:** subconjunto dos comandos usados para inserir, atualizar e apagar dados.

Durante este conteúdo, abordaremos os principais comandos para realizar as ações de:

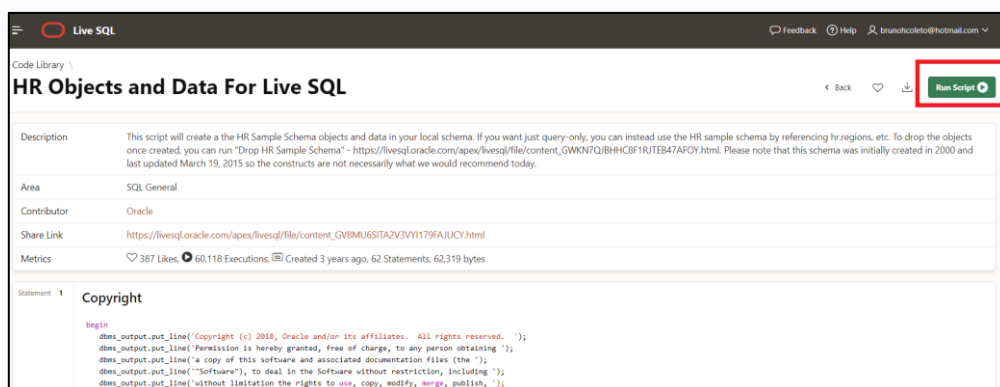
- **Select:** Buscar e Exibir dados das tabelas;
- **Insert:** Inserir dados em tabelas;
- **Update:** Atualizar dados em tabelas;
- **Delete:** Deletar dados em tabelas.

Para realizarmos estas ações, utilizaremos, como exemplo, o Banco de Dados Oracle. Atualmente a Oracle disponibiliza o acesso *web* a um ambiente para estudo e aperfeiçoamento, totalmente gratuito. Este ambiente permitirá que façamos a utilização dos comandos propostos nesta disciplina.

Para isso, utilize o tutorial abaixo:

- Acesse o endereço:
https://livesql.oracle.com/apex/livesql/file/content_GV8MU6SITA2V3VYI179FAJUCY.html;
- Realize seu cadastro de forma simples no *site* da Oracle;
- Clique em *Run Script*, conforme a imagem abaixo.

Figura 1 – Página inicial do Live SQL

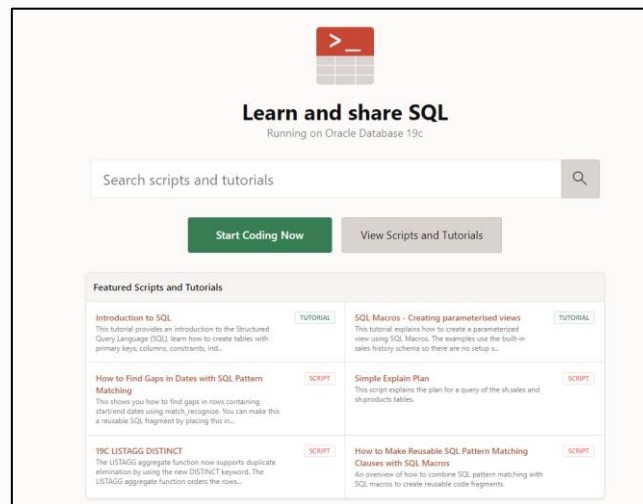


Fonte: Oracle – Live SQL

Em seguida, clique em **Start Coding Now**

A Oracle oferece tutoriais como sugestões para codificação, facilitando assim a vida dos usuários.

Figura 2 – Tutorial da Oracle

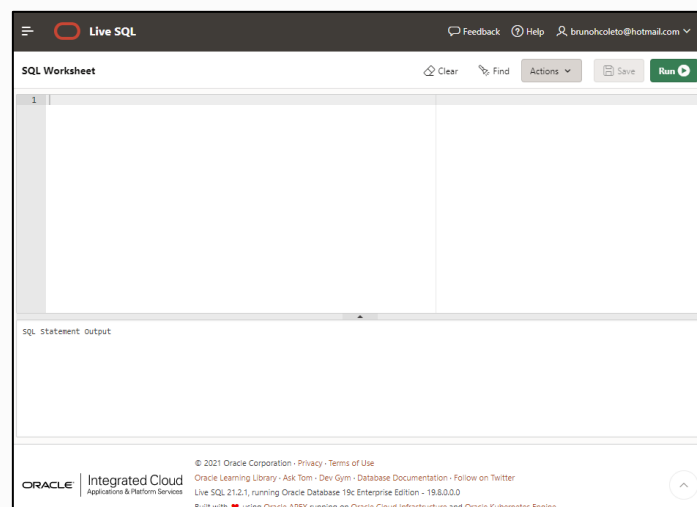


Fonte: Oracle – Live SQL

Seremos direcionados para o ambiente em que iremos trabalhar para execução dos *scripts* no ambiente Oracle.

A lacuna superior será o local onde iremos inserir os *scripts* e logo abaixo o resultado será demonstrado. Para execução de um comando SQL, utilize o botão **Run**.

Figura 3 – Execução de um comando SQL

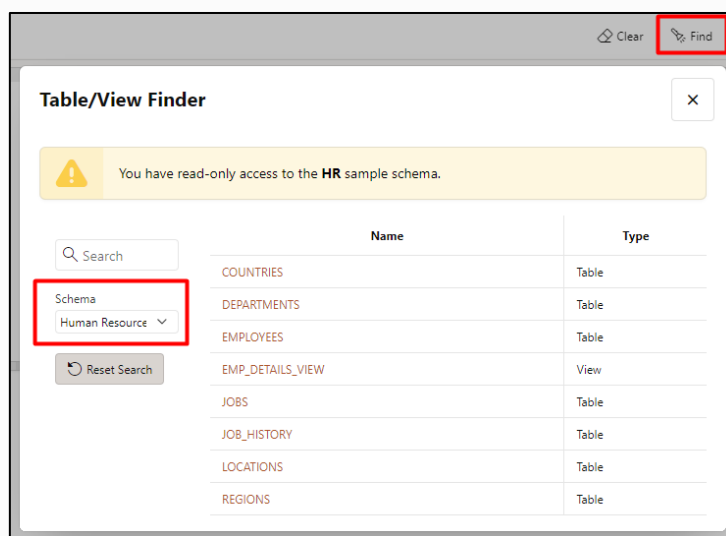


Fonte: Oracle – Live SQL

Observações importantes:

- Por padrão, você estará trabalhando no seu *Schema*. Entretanto, não temos nenhuma tabela criada em nosso *Schema*. Deste modo, selecionaremos um *Schema* padrão do Banco de Dados Oracle;
- Clique em **Find**, selecione o *Schema*: **Human Resource HR**;
- Logo, será demonstrada uma mensagem que temos o acesso de somente leitura em torno do esquema HR.

Figura 4 – Table/View Finder



Fonte: Oracle – Live SQL

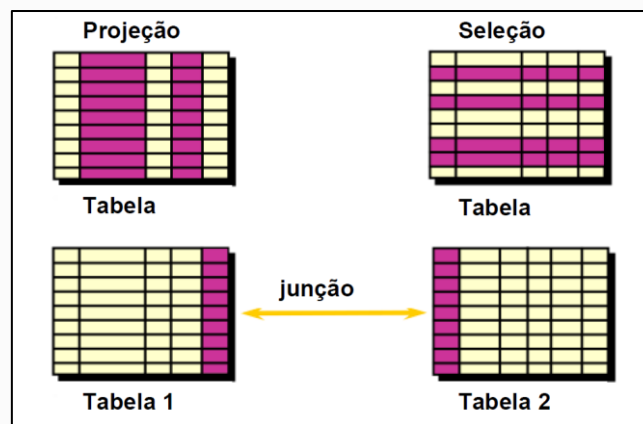
Select

A instrução *SELECT* tem por objetivo recuperar/consultar informações do banco de dados.

Com um comando SQL, podemos realizar as seguintes ações:

- **Projeção:** escolher quais colunas de determinadas tabelas devem ser retornadas por uma consulta e também escolher a quantidade de colunas a serem exibidas;
- **Seleção:** escolher quais linhas de uma tabela podem ser retornadas e também utilizar vários critérios para restringir as linhas a serem exibidas;
- **Junção:** realizar a junção de dados armazenados em várias tabelas e unificá-los para demonstração de um único resultado.

Figura 5 – Projeção, seleção e junção de tabelas



Fonte: o autor (2021)

De modo simplificado, um comando *Select* possui duas cláusulas importantes.

1. *SELECT*: onde especificamos as colunas a serem exibidas na consulta;
2. ***: Selecionar todas colunas da tabela;
3. *Distinct*: suprimir itens duplicados;
4. Apelido: cabeçalhos para separação de diferentes colunas;
5. Cláusula *FROM*: onde especificamos uma ou mais tabelas que contêm colunas listas na cláusula *SELECT*.

Exemplo de Sintaxe básica:

```
SELECT *
```

```
FROM hr.departments;
```

Explicação do comando:

Selecione todos os campos da tabela *departments*.

Obs.: observem que estamos utilizando a nomenclatura **hr** antes da tabela *departments*. Isso se faz necessário pois na estrutura do Oracle é necessário informar qual *schema* determinada tabela está alocada.

Logo abaixo compartilho a execução deste comando no ambiente Oracle.

Figura 6 – Comando *Select*

SQL Worksheet			
1	<i>SELECT</i> *		
2	FROM hr.departments;		

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury	-	1700
130	Corporate Tax	-	1700
140	Control And Credit	-	1700

Fonte: Oracle – Live SQL

Também seria uma possibilidade selecionar todas as colunas da tabela, informando na cláusula *select*.

Figura 7 – Cláusula *select*

SQL Worksheet			
1	<i>SELECT</i> DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, LOCATION_ID		
2	FROM hr.departments;		

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500

Fonte: Oracle – Live SQL

Selecionando colunas específicas

Podemos utilizar o comando *Select* para retornar colunas específicas de determinada tabela, especificando o nome das colunas separados por vírgula. O exemplo abaixo exhibe os campos de identificação para locação e departamento da tabela *Departments*.

SELECT location_id, department_id

FROM departments

Figura 8 - Campos de identificação para locação e departamento da tabela *Departments*

SQL Worksheet	
1	SELECT location_id, department_id
2	FROM hr.departments
LOCATION_ID	DEPARTMENT_ID
1700	10
1800	20
1700	30
2400	40
1500	50
1400	60
2700	70
2500	80
1700	90
1700	100
1700	110
1700	120

Fonte: Oracle – Live SQL

Selecionando itens sem duplicação (*DISTINCT*)

Durante a execução de instruções SQL nos deparamos com a apresentação de resultados repetidos.

No exemplo abaixo, demonstraremos o incidente e também a utilização da cláusula *Distinct* para resolução do caso de forma simplificada.

```
select DEPARTMENT_ID  
from hr.EMPLOYEES  
order by DEPARTMENT_ID;
```

Figura 9 - Utilização da cláusula *Distinct*

SQL Worksheet

```
1 select DEPARTMENT_ID from hr.EMPLOYEES order by DEPARTMENT_ID;
```

DEPARTMENT_ID
10
20
20
30
30
30
30
30
30
30
40
50
50
50
50
50
50
50
50
50

Fonte: Oracle – Live SQL

Para resolvermos este caso, utilizaremos a inclusão da Cláusula *DISTINCT*.

```
select DISTINCT DEPARTMENT_ID  
from hr.EMPLOYEES  
order by DEPARTMENT_ID;
```

Figura 10 – Cláusula *Distinct*

SQL Worksheet

```
1 select DISTINCT DEPARTMENT_ID from hr.EMPLOYEES order by DEPARTMENT_ID;
```

DEPARTMENT_ID
10
20
30
40
50
60
70
80
90
100
110
-

[Download CSV](#)
12 rows selected.

Fonte: Oracle – Live SQL

Observe com o resultado exibido, que não houve um único registro com o mesmo código de departamento utilizando a cláusula *Distinct*.

Order By (Ordenando)

A expressão *ORDER BY* pode ser utilizada para ordenação dos registros a serem exibidos.

De forma padrão, apenas se inserirmos *Order BY* e a coluna na qual desejamos ordenar é que os registros serão exibidos de forma ascendente. Entretanto, se desejamos realizar a ordenação de modo decrescente, devemos inserir a cláusula *DESC*.

Um exemplo pode ser visto abaixo.

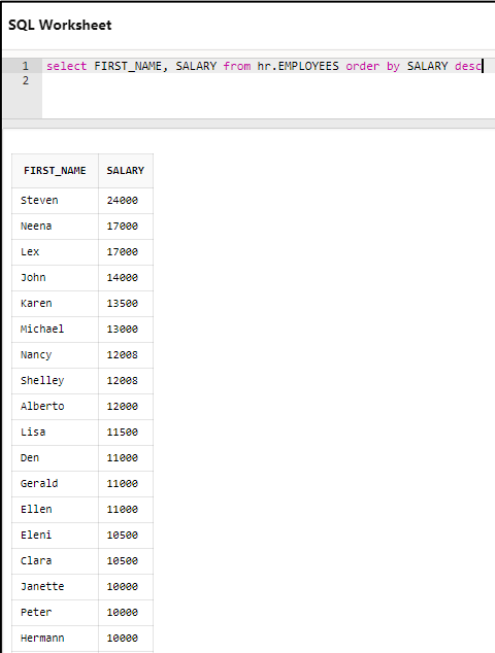
Selecionaremos o nome e salário da tabela *Employees*, ordenando o salário de forma decrescente.

```
select FIRST_NAME, SALARY
```

```
from EMPLOYEES
```

```
order by SALARY desc
```

Figura 11 - Ordenando o salário de forma decrescente



FIRST_NAME	SALARY
Steven	24000
Neena	17000
Lex	17000
John	14000
Karen	13500
Michael	13000
Nancy	12000
Shelley	12000
Alberto	12000
Lisa	11500
Den	11000
Gerald	11000
Ellen	11000
Eleni	10500
Clara	10500
Janette	10000
Peter	10000
Hermann	10000

Fonte: Oracle – Live SQL

Se retirarmos a cláusula *DESC* ao final da consulta, o salário será exibido de forma ascendente.

Figura 12 – Exibição do salário de forma ascendente

SQL Worksheet	
1	<code>select FIRST_NAME, SALARY from hr.EMPLOYEES order by SALARY;</code>
2	

FIRST_NAME	SALARY
TJ	2100
Steven	2200
Hazel	2200
James	2400
KI	2400
Karen	2500
James	2500
Joshua	2500
Peter	2500
Martha	2500
Randall	2500
Guy	2600
Randall	2600
Donald	2600

Fonte: Oracle – Live SQL

Restrições

Existem vários operadores que podem ser usados para a restrição do número de resultados.

Operadores de comparação:

- Igualdade: =
- Diferença: <>
- Demais: > < >= <=

Outros operadores:

- *IN*
→ *NOT IN*
- *BETWEEN*
- *EXISTS*
- *IS NULL*
→ *IS NOT NULL*
- *LIKE*

Lógicos:

- *AND OR NOT*
→ A utilização dos operadores lógicos utiliza a precedência de parênteses como em linguagens de programação.

Um exemplo de uso da igualdade é a busca dos empregados de um determinado departamento, por exemplo:

```
select * from
```

```
EMPLOYEES where
```

```
DEPARTMENT_ID = 90
```

Figura 13 - Busca dos empregados de um determinado departamento

SQL Worksheet										
1	select * from hr.EMPLOYEES where DEPARTMENT_ID = 90									
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-	100	90

[Download CSV](#)
3 rows selected.

Fonte: Oracle – LIVE SQL

Um outro exemplo, é buscarmos um funcionário cujo o sobrenome, termina com (King).

Exemplo:

```
select * from hr.EMPLOYEES
```

```
where last_name like '%King'
```

Figura 14 – Busca por sobrenome

SQL Worksheet										
1	select * from hr.EMPLOYEES where last_name like '%King'									
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
156	Janette	King	JKING	011.44.1345.429268	30-JAN-04	SA_REP	10000	.35	146	80

[Download CSV](#)
2 rows selected.

Fonte: Oracle – Live SQL

E por fim, realizaremos uma pesquisa, para demonstrar quais os funcionários que recebem salário maior que 13.000.

```
select *
```

```
from hr.EMPLOYEES
```

where SALARY > 13000;

Figura 15 – Pesquisa para demonstrar os funcionários que recebem mais que 13.000

SQL Worksheet										
1 select * from hr.EMPLOYEES where SALARY > 13000;										
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-	100	90
145	John	Russell	JRUSSEL	011.44.1344.429268	01-OCT-04	SA_MAN	14000	.4	100	80
146	Karen	Partners	KPARTNER	011.44.1344.467268	05-JAN-05	SA_MAN	13500	.3	100	80
Download CSV 5 rows selected.										

Fonte: Oracle – Live SQL

Operadores lógicos:

Os operadores lógicos têm por objetivo combinar os resultados de duas condições de comparação para produzir um único resultado (*AND* e *OR*) ou para inverter o resultado de uma simples comparação (*NOT*).

- *NOT*: às vezes, no português falado dizemos, por exemplo, que queremos todos os funcionários que **não** estão no departamento 30.

A consulta abaixo faz exatamente isto:

select *

from hr.EMPLOYEES

where not DEPARTMENT_ID = 30

Figura 16 – Consulta dos funcionários que **não** estão no departamento 30

SQL Worksheet										
1 select * from hr.EMPLOYEES where not DEPARTMENT_ID = 30;										
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	4800	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4500	05-FEB-06	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200	-	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-02	FI_MGR	12000	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-02	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	8200	-	108	100
111	Ismael	Scliarra	ISCLARRA	515.124.4369	30-SEP-05	FI_ACCOUNT	7700	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	07-MAR-06	FI_ACCOUNT	7800	-	108	100
113	Luis	Popp	LOPP	515.124.4567	07-DEC-07	FI_ACCOUNT	6900	-	108	100

Fonte: Oracle – Live SQL

Ou também:

```
select * from hr.EMPLOYEES where DEPARTMENT_ID <> 30;
```

Figura 17 – Segundo exemplo da mesma consulta

SQL Worksheet										
1 select * from hr.EMPLOYEES where DEPARTMENT_ID <> 30;										
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	4900	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200	-	103	60

Fonte: Oracle – Live SQL

- **AND:** às vezes, no português falado dizemos, por exemplo, que queremos todos os funcionários que estão no departamento 30 e que ganham mais de 3.000.

Vejamos o exemplo abaixo:

```
select *
```

```
from hr.EMPLOYEES
```

```
where DEPARTMENT_ID = 30 AND SALARY > 3000;
```

Figura 18 - Seleção dos funcionários que estão no departamento 30 e ganham mais de 3.000

SQL Worksheet										
1 select * from hr.EMPLOYEES where DEPARTMENT_ID = 30 AND SALARY > 3000;										
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
114	Den	Raphaely	DRAPHEAL	515.127.4561	07-DEC-02	PU_MAN	11000	-	100	30
115	Alexander	Khoo	AKHOO	515.127.4562	18-MAY-03	PU_CLERK	3100	-	114	30

Download CSV
2 rows selected.

Fonte: Oracle – Live SQL

- **OR:** existem outros casos que queremos todos os funcionários que estão no departamento 30 ou que ganham mais de 17.000.

Vejamos o exemplo abaixo:

```
select *
from hr.EMPLOYEES
where DEPARTMENT_ID = 30 OR SALARY > 17000;
```

Figura 19 – Seleção dos funcionários que estão no departamento 30 ou que ganham mais de 17.000

SQL Worksheet

```
1 select * from hr.EMPLOYEES where DEPARTMENT_ID = 30 OR SALARY > 17000;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
114	Den	Raphaely	DRAPHEAL	515.127.4561	07-DEC-02	PU_MAN	11000	-	100	30
115	Alexander	Khoo	AKHOO	515.127.4562	18-MAY-03	PU_CLERK	3100	-	114	30
116	Shelli	Baida	SBAIDA	515.127.4563	24-DEC-05	PU_CLERK	2900	-	114	30
117	Sigal	Tobias	STOBIAS	515.127.4564	24-JUL-05	PU_CLERK	2800	-	114	30
118	Guy	Himuro	GHIMURO	515.127.4565	15-NOV-06	PU_CLERK	2600	-	114	30
119	Karen	Colmenares	KCOLMENA	515.127.4566	10-AUG-07	PU_CLERK	2500	-	114	30

Download CSV
7 rows selected.

Fonte: Oracle – Live SQL

- IN: para retornar os usuários que estão em uma determinada lista de departamentos, utilizamos:

```
select *
from EMPLOYEES
where DEPARTMENT_ID IN (90,100);
```

Figura 20 - Retornar os usuários que estão em uma determinada lista de departamentos

SQL Worksheet

```
1 select * from hr.EMPLOYEES where DEPARTMENT_ID IN (90,100);
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-	100	90
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-02	FI_MGR	12000	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-02	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	8200	-	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	30-SEP-05	FI_ACCOUNT	7700	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	07-MAR-06	FI_ACCOUNT	7800	-	108	100
113	Luis	Popp	LPPOP	515.124.4567	07-DEC-07	FI_ACCOUNT	6900	-	108	100

Download CSV
9 rows selected.

Fonte: Oracle – Live SQL

- **NOT IN:** para retornar os usuários que **não** estão nos departamentos de uma determinada lista, utilizamos:

select *

from hr.EMPLOYEES

where DEPARTMENT_ID NOT IN (10,30,50);

Figura 21 - Retornar os usuários que **não** estão nos departamentos de uma determinada lista

1	select *
2	from hr.EMPLOYEES
3	where DEPARTMENT_ID NOT IN (10,30,50);
4	

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	4800	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200	-	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-02	FI_MGR	12008	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-02	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	8200	-	108	100

Fonte: Oracle – Live SQL

- **BETWEEN:** o operador *between* permite testar se um determinado campo assume o valor dentro de um intervalo especificado. É utilizado por ser mais prático que o teste de >= e <=. Para retornar os usuários que recebem entre 5.000 e 10.000, utilizamos:

select *

from hr.EMPLOYEES

where SALARY between 5000 and 10000

Figura 22 - Retornar os usuários que recebem entre 5.000 e 10.000

1	select * from hr.EMPLOYEES where SALARY between 5000 and 10000
2	

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	-	103	60
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-02	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	8200	-	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	30-SEP-05	FI_ACCOUNT	7700	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	07-MAR-06	FI_ACCOUNT	7800	-	108	100
113	Luis	Popp	LPOPP	515.124.4567	07-DEC-07	FI_ACCOUNT	6900	-	108	100
120	Matthew	Weiss	MWEISS	650.123.1234	18-JUL-04	ST_MAN	8000	-	100	50
121	Adam	Frippe	AFRIPP	650.123.2234	10-APR-05	ST_MAN	8200	-	100	50
122	Payan	Kaufling	PKAUFLIN	650.123.3234	01-MAY-03	ST_MAN	7900	-	100	50
123	Shanta	Vollman	SVOLLMAN	650.123.4234	10-OCT-05	ST_MAN	6500	-	100	50
124	Kevin	Hourgos	KHOURGOS	650.123.5234	16-NOV-07	ST_MAN	5800	-	100	50
150	Peter	Tucker	PTUCKER	011.44.1344.129268	30-JAN-05	SA_REP	10000	.3	145	80
151	David	Bernstein	DBERNSTE	011.44.1344.345268	24-MAR-05	SA_REP	9500	.25	145	80

Fonte: Oracle – Live SQL

- *IS NULL* e *IS NOT NULL*: para encontrar valores nulos, você deve utilizar o operador *IS NULL*. Os operadores = ou <> **não** funcionam com valores nulos.

Para retornar os usuários que não estão lotados em nenhum departamento, utilizamos:

```
select *
from EMPLOYEES
where DEPARTMENT_ID is NULL;
```

Figura 23 - Retornar os usuários que não estão lotados em nenhum departamento

1	select * from hr.EMPLOYEES where DEPARTMENT_ID is NULL;	
2		

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-07	SA_REP	7000	.15	149	-

[Download CSV](#)

Fonte: Oracle – Live SQL

Caso quisermos selecionar os que não estejam lotados em um departamento, podemos usar:

```
select *
from hr.EMPLOYEES
where DEPARTMENT_ID is not NULL;
```

Figura 24 - Selecionar os que não estejam lotados em um departamento

1	select * from hr.EMPLOYEES where DEPARTMENT_ID is not NULL;	
---	---	--

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	4800	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200	-	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-02	FI_MGR	12008	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-02	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	8200	-	108	100

Fonte: Oracle – Live SQL

- *LIKE*: utilizado quando se deseja obter colunas de um registro que sigam um padrão especificado. Exemplo:

Selecione os empregados cujo nome inicia-se com 'Alex':

```
select *  
from EMPLOYEES  
where FIRST_NAME like 'Alex%'
```

Figura 25 - Empregados cujo nome inicia-se com 'Alex'

SQL Worksheet										
1 select * from hr.EMPLOYEES where FIRST_NAME like 'Alex%'										
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
185	Alexis	Bull	ABULL	650.509.2876	20-FEB-05	SH_CLERK	4100	-	121	50
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	-	102	60
115	Alexander	Khoo	AKHOO	515.127.4562	18-MAY-03	PU_CLERK	3100	-	114	30

Download CSV
3 rows selected.

Fonte: Oracle – Live SQL

Selecione os empregados cujo nome possua a letra 'o' na 2ª posição:

```
select *  
from EMPLOYEES  
where FIRST_NAME like '_o%'
```

Figura 26 - Selecione os empregados cujo nome possua a letra 'o' na 2ª posição

SQL Worksheet										
1 select * from hr.EMPLOYEES where FIRST_NAME like '_o%'										
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	8200	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	07-MAR-06	FI_ACCOUNT	7800	-	108	100
130	Mozhe	Atkinson	MATKINSO	650.124.6234	30-OCT-05	ST_CLERK	2800	-	121	50
139	John	Seo	JSEO	650.121.2019	12-FEB-06	ST_CLERK	2700	-	123	50
140	Joshua	Patel	JPADEL	650.121.1834	06-APR-06	ST_CLERK	2500	-	123	50
145	John	Russell	JRUSSEL	011.44.1344.429268	01-OCT-04	SA_MAN	14000	.4	100	80
160	Louise	Doran	LDORAN	011.44.1345.629268	15-DEC-05	SA_REP	7500	.3	146	80
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-06	SA_REP	8600	.2	149	80
198	Donald	OConnell	DOCONNEL	650.507.9833	21-JUN-07	SH_CLERK	2600	-	124	50
199	Douglas	Grant	DGRANT	650.507.9844	13-JAN-08	SH_CLERK	2600	-	124	50

Download CSV
10 rows selected.



VIDEOAULA 01

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda sobre *select* e os operadores lógicos.



Instruções DML

As instruções DML (*Data Manipulation Language*) são um subconjunto do SQL que tem por objetivo manipular dados.

Durante esta aula, trabalharemos com os comandos *Insert* (inserir linhas em uma tabela), *Update* (alterar valores em uma tabela) e por fim o *Delete* (remover linhas de determinada tabela).

Insert (inserção)

O comando *Insert* é utilizado para adicionar novas linhas em uma tabela obedecendo a seguinte sintaxe:

- Tabela: nome da tabela;
- Coluna: nome da coluna da tabela a ser preenchida e;
- Valor: Valor correspondente para a coluna.

A instrução *Insert* com a cláusula *Values* adiciona somente uma linha por vez a uma tabela.

Por exemplo:

```
INSERT INTO departments (department_id, department_name,  
manager_id, location_id)
```

```
VALUES (70, 'Public Relations', 100, 1700);
```

Neste exemplo, estamos inserindo um novo registro na tabela *Departaments*.

Figura 27 - Inserindo um novo registro na tabela *Departaments*

SQL Worksheet

```
1 INSERT INTO hr.departments (department_id, department_name,
2 manager_id, location_id)
3 VALUES (70, 'Public Relations', 100, 1700);
4
5
6
7 select * from hr.departments;
```

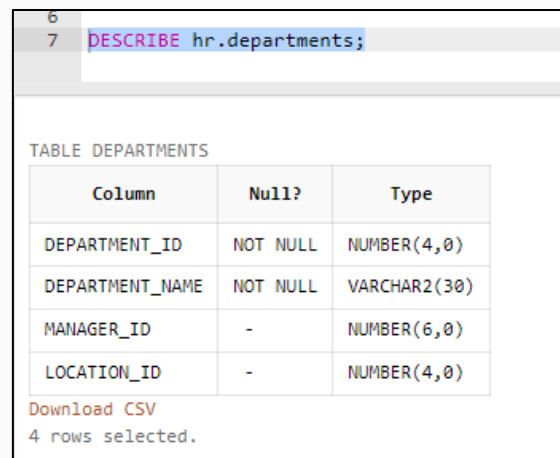
DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700

Fonte: Oracle – Live SQL

Para que o usuário possa visualizar os nomes das colunas e os seus respectivos tipos de dados de determinada tabela, o usuário pode utilizar o seguinte comando:

DESCRIBE hr.departments;

Figura 28 – Utilização do comando “DESCRIBE hr.departments”



The screenshot shows a SQL query editor with the command 'DESCRIBE hr.departments;' entered. Below the query, the results are displayed as a table with the title 'TABLE DEPARTMENTS'. The table has three columns: 'Column', 'Null?', and 'Type'. It contains four rows of data for the departments table. Below the table, there is a link to 'Download CSV' and a message stating '4 rows selected.'

Column	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4,0)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID	-	NUMBER(6,0)
LOCATION_ID	-	NUMBER(4,0)

[Download CSV](#)
4 rows selected.

Fonte: Oracle – Live SQL



VIDEOAULA 02

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda sobre *insert*.



Update (Alteração)

O comando *update* é utilizado para modificar linhas existentes em tabelas no banco de dados.

Por exemplo:

O comando abaixo realiza uma alteração multiplicando o salário de cada colaborador por 1.1.

```
update hr.EMPLOYEES
```

```
set SALARY = 1.1*SALARY;
```

Figura 29 – Comando que realiza uma alteração multiplicando o salário de cada colaborador por 1.1

SQL Worksheet

1

update hr.EMPLOYEES set SALARY = 1.1*SALARY;

2

3

select * from hr.EMPLOYEES

ORA-01031: insufficient privileges

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	4800	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200	-	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-02	FI_MGR	12008	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-02	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	8200	-	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	30-SEP-05	FI_ACCOUNT	7700	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	07-MAR-06	FI_ACCOUNT	7800	-	108	100

Fonte: Oracle – Live SQL

Delete (Deleção)

O comando delete tem por objetivo remover linhas de uma tabela. Por exemplo: o comando abaixo realiza a exclusão de todos os países.

```
Delete
from COUNTRIES;
```

Figura 30 – Utilização do comando Delete

SQL Worksheet		
1	delete from hr.COUNTRIES;	
2		
3	select * from hr.COUNTRIES	
COUNTRY_ID	COUNTRY_NAME	REGION_ID
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2
CA	Canada	2
CH	Switzerland	1
CN	China	3
DE	Germany	1
DK	Denmark	1
EG	Egypt	4
FR	France	1

Fonte: Oracle – Live SQL

Um outro exemplo:

O comando abaixo, exclui o país com código UR:

Delete

from COUNTRIES

where COUNTRY_ID = 'UR';

Figura 31 – Outro exemplo do comando Delete

SQL Worksheet

1

2

3

delete from COUNTRIES where COUNTRY_ID = 'UR';

select * from hr.COUNTRIES |

COUNTRY_ID	COUNTRY_NAME	REGION_ID
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2
CA	Canada	2
CH	Switzerland	1
CN	China	3
DE	Germany	1
DK	Denmark	1
EG	Egypt	4
FR	France	1
IL	Israel	4
IN	India	3
IT	Italy	1
JP	Japan	3
KW	Kuwait	4
ML	Malaysia	3
MX	Mexico	2
NG	Nigeria	4
NL	Netherlands	1
SG	Singapore	3
UK	United Kingdom	1

Fonte: Oracle – Live SQL.



VIDEOAULA 03

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda sobre *update* e *delete*.



INDICAÇÃO DE LEITURA

Leia o livro *“Introdução a Linguagem SQL: Tradução da 5a Edição”*, por Thomas Nield para conhecer mais sobre o assunto relacionado à introdução de banco de dados.

INDICAÇÃO DE VÍDEO

Agora, assista ao vídeo que aborda sobre a introdução de linguagem SQL. Disponível em: <https://www.youtube.com/watch?v=loD-6CpgU1E>. Acesso em: 6 jul. 2021.

CURIOSIDADES

No site oficial da Oracle você encontra mais informações sobre comandos SQL. *Link* para acesso: Disponível em: <https://livesql.oracle.com/apex/f?p=590:1:7803975526589::NO:RP:n>. Acesso em: 6 jul. 2021.

Aula 02 - Detalhamento sobre a estrutura geral, meios de armazenamento, organização de registros

No mundo real em que vivemos, devemos considerar diversos aspectos importantes quando pensamos em como os dados são armazenados em um banco de dados e definidos em forma de registro. No decorrer desta aula falaremos a respeito de como é a arquitetura de banco de dados e suas vantagens.

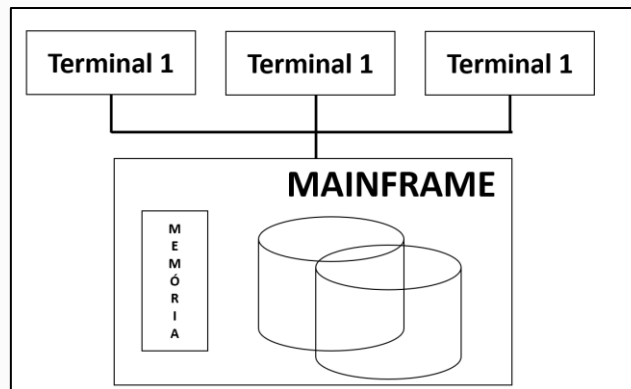
Arquitetura Centralizada

Segundo França (2015), quando utilizamos o termo relacionado a banco de dados, o dicionário Michaelis define que a arquitetura em camadas, como o projeto de um sistema de computador em camadas, de acordo com a função ou propriedade.

França (2015) relata que as arquiteturas de banco de dados centralizados utilizavam o conceito de mainframes, centralizando o processamento de funções e programas de aplicação, interface e diversas funcionalidades relacionadas ao SBGD. De modo simplificado, os usuários acessavam seus terminais de acesso para extração de resultados e todo o processamento era executado no mainframe.

A Figura 1 ilustra esse processamento.

Figura 1 - Arquitetura Centralizada



Fonte: o autor (2021)

Até chegarmos nos dias atuais, houveram algumas mudanças quanto aos preços de equipamentos de informática (Hardware e Software) ou seja, aconteceu uma grande queda de preços, facilitando assim que os usuários substituíssem antigos terminais por computadores e estações de trabalho. Durante essa mudança, cientistas buscaram viabilizar o poder computacional para os usuários. De modo simplificado, esse processo desencadeou uma arquitetura mais conhecida hoje em dia, chamada de Cliente – Servidor.

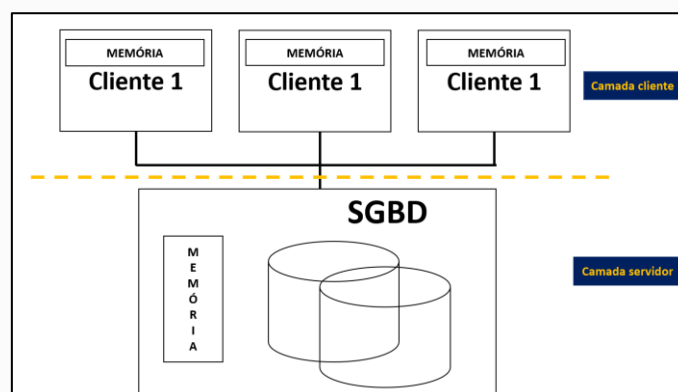
Arquitetura Cliente-Servidor de Duas Camadas

Segundo França (2015), a arquitetura cliente servidor de duas camadas é uma evolução da arquitetura centralizada, devido à substituição de terminais antigos por estações de trabalho, aumentando o poder computacional do lado cliente, possibilitando que o processamento para eles “desafogue” os servidores.

De modo geral, um cliente é geralmente uma máquina com poder computacional e diversas funcionalidades de interfaces. Embora, sempre se faz necessário que o cliente realize a conexão com o banco e o servidor no qual viabiliza o acesso ao banco de dados.

A Figura 2 ilustra este processo de modo simplificado.

Figura 2 - Arquitetura Cliente/Servidor de duas camadas



Fonte: o Autor (2021)

Nesta arquitetura, o primeiro passo é estabelecer uma conexão entre o cliente-servidor e SGBD. Em seguida, o cliente solicita, por meio de comandos, determinadas consultas de dados, as informações são processadas no servidor e retornadas ao cliente.

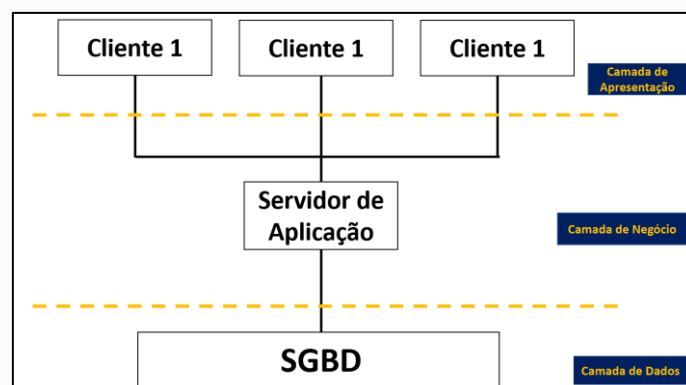
Arquitetura Cliente-Servidor de Três Camadas

França (2015) relata que com o crescimento da WWW (World Wide Web – Rede de Alcance Mundial – ou simplesmente Web) e as responsabilidades da arquitetura cliente-servidor de duas camadas sofreram mudanças que levaram a arquitetura cliente-servidor de duas camadas a uma evolução denominada arquitetura cliente-servidor de três camadas, ou arquitetura de três camadas. Na arquitetura de três camadas foi alocada uma camada intermediária entre a máquina cliente e o servidor de banco de dados. Essa camada interlocutora passou a chamar-se de servidor de aplicação ou servidor Web, dependendo do contexto onde o aplicativo está inserido.

Elmasri e Navathe (2005) reprisam que essa camada intermediária passou a desempenhar um papel fundamental para armazenamento de toda a regra de negócio, utilizadas para acesso de dados referente ao servidor de banco de dados.

A Figura 3 demonstra de forma simplificada a evolução para a arquitetura cliente / servidor de três camadas.

Figura 3 - Arquitetura Cliente/Servidor de três camadas



Fonte: o autor (2021)

Para melhor explicação desta camada, separamos de modo simplificado a explicação de cada subcamada, organizando da seguinte forma:

- Camada de Apresentação (Cliente): implementações de interface com usuário. França (2015) define essa subcamada como (GUI - Graphics User Interface – Interface Gráfica para o Usuário), tendo como papel específico a interação com o usuário em torno das solicitações e em seguida apresentação dos resultados;
- Camada de Negócio (Servidor Aplicação): implementações de regra de negócio. França (2015) define que essa subcamada tem por objetivo a não interação diretamente com o usuário, mas sim recebendo e retornando suas solicitações (Camada de Apresentação), realizando o processamento e enviando ou não para a camada subsequente (Camada de Dados);

- Camada de Dados: (Servidor de Dados): Repositório de dados (Servidor). França (2015) informa que essa subcamada recebe e executa as solicitações providas da camada de negócio e a retornando com os dados processados. Vale lembrar, que a transição dos dados até a subcamada apresentação é de responsabilidade da subcamada de negócio.

No ponto de vista consolidado, a arquitetura de três camadas, pode ser semelhante a arquitetura de duas camadas. Entretanto, ao excluirmos do servidor as regras de negócio, ocasionalmente o servidor será menos sobrecarregado, pois seu papel acaba sendo o gerenciamento de dados e não interlocução direta com o usuário.



VIDEOAULA 01

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda sobre a arquitetura centralizada, arquitetura cliente servidor de duas e três camadas.

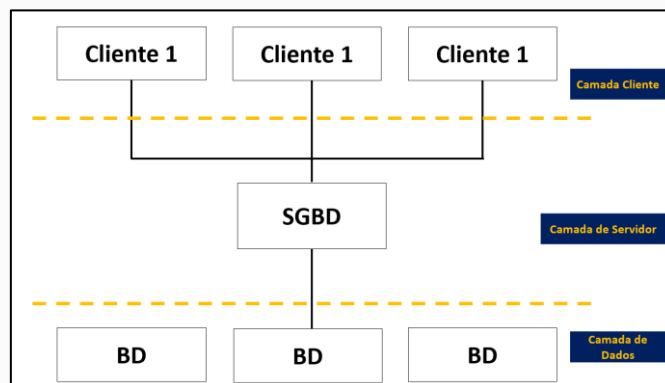


Arquitetura Distribuída

A arquitetura distribuída fornece grandes vantagens para o banco de dados, assim como algumas dificuldades interligadas a gerenciamento. Elmasri e Navathe (2005) reprisam que funções comuns de gerenciamento do banco de dados [...] não se aplicam, contudo, a esse cenário da arquitetura Distribuída.

Quando imaginamos a questão de distribuição, logo pensamos em vários bancos de dados descentralizados em diferentes localizações, mas em que alguns momentos se unificam. De modo simplificado, França (2015) define que um BDD (Banco de Dados Distribuído) é um conjunto de banco de dados distribuídos por meio de uma rede de computadores, mas logicamente inter-relacionados, enquanto o SGBDD (Sistema Gerenciador de Banco de dados Distribuído) não apenas gerencia o BDD, mas também torna a distribuição e transações transparentes para o usuário. A Figura 4 exemplifica melhor este cenário.

Figura 1 - Arquitetura Distribuída



Fonte: o autor (2021)

Dentro desta arquitetura, temos dois tipos relacionados a banco de dados distribuídos:

- Homogêneos: um único tipo de Banco de Dados;
- Heterógenos: mais de um tipo de Banco de Dados.

Arquitetura Distribuída - Vantagens

França (2015) informa que algumas vantagens relacionadas à arquitetura distribuída são: (i) descentralização dos dados, aumentando o poder computacional de processamento; a (ii) fragmentação dos dados levando em consideração a estrutura organizacional, persistindo os dados no local desejado (Ex.: Departamento de Compras) aumentando a autonomia local; a (iii) melhoria no desempenho devido à proximidade dos dados, paralelismo e balanceamento entre os servidores de dados; (iv) tolerância a falhas aumentando a disponibilidade dos dados; (v) economia na aquisição de servidores menores a medida que o poder computacional exigido for aumentado; (vi) facilidade de acrescentar ou remover novos servidores.

O Gerenciamento de Dados distribuídos permite a transparência e eficiência de distribuição dos dados, mas ao mesmo tempo a transparência de fragmentação destes dados pode ocorrer.



VIDEOAULA 02

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda sobre a Arquitetura Distribuída e suas vantagens.



Arquitetura Distribuída - Desvantagens

França (2015) informa que algumas desvantagens relacionadas à arquitetura distribuída são: o (i) complexidade exigida para garantir a distribuição de forma transparente para o usuário; (ii) custo maior de implementação devido ao trabalho extra exigido; (iii) planejamento mais difícil devido à fragmentação, alocação e, algumas vezes, a replicação dos dados; (iv) integridade do banco exige alto recurso de rede; (v) exigência de maior segurança tanto nos servidores quanto na infraestrutura; (vi) inexistência de um padrão que auxilie a conversão dos bancos de dados centralizados para os banco de dados distribuídos; (vii) poucos casos práticos disponíveis para serem analisados.

Elmasri e Navathe (2005) retrata que a obtenção das vantagens do banco de dados distribuído leva ao projeto e à implementação de um sistema gerenciador mais complexo, no qual o SGBDD deve prover, além das funcionalidades do SGBD centralizado, (i) rastreamento de dados; (ii) processamento de consultas distribuídas; (iii) gerenciamento de transações distribuídas; (iv) gerenciamento dos dados replicados; (v) recuperação de banco de dados distribuído; (vi) segurança; (vii) gerenciamento do diretório distribuído.

Arquitetura Distribuída – Fragmentação de Dados

A fragmentação de banco de dados é resumidamente uma partição de dados que se divide em partes menores proporcionando um gerenciamento mais rápido e fácil. Um exemplo mais

simplificado em torno da fragmentação de dados é reprisado por (FRANÇA, 2015) onde informa a fragmentação com relação inteira, ou seja, os dados de uma tabela inteira são colocados em um único servidor do BDD. Dessa maneira, sempre que se precisar de alguma informação da tabela, o SGBDD irá busca essa informação no servidor que a mantém.

Um exemplo de fragmentação é demonstrado na tabela 1. No qual, devemos imaginar uma empresa com cinco servidores de dados e diversas tabelas distintas.

Tabela 1 – Fragmentação de Dados

Servidor	Tabela	Qtde Registros
S01	Cliente	15000
S02	Fornecedor	200
S03	Compra	55000
S04	Compra_item	125000
S05	Estoque	5000

Fonte: o autor (2021)

- Servidor 01: Armazenamento de dados relacionados a Clientes;
- Servidor 02: Armazenamento de dados relacionados a Fornecedores;
- Servidor 03: Armazenamento de dados relacionados a Compras. Exemplo: Dados de Compra, qual cliente efetuou a compra, valor e demais formas de pagamento;
- Servidor 04: Armazenamento de dados relacionados aos Itens de Compras. Exemplo: Quantidade de cada item, valor pago, qual cliente adquiriu determinado item, valor de compra, valor de venda;
- Servidor 05: Armazenamento de dados relacionados aos Itens de Estoque.

Neste nosso exemplo, o Servidor 02 irá necessitar de um poder computacional (*Hardware e Software*) inferior ao servidor 04. Deste modo, acontece um desbalanceamento de cargas entre todos os servidores.

Para resolver este caso de modo simplificado, existe a Fragmentação Horizontal. No qual, basicamente, os registros são divididos entre vários servidores de modo horizontal. Como pode ser observado na Tabela 2, cada registro é alocado em um único servidor em conjunto com cada coluna.

Tabela 2 - Fragmentação Horizontal

Servidor	ID_Cliente	Nome_Cliente	Cidade_Cliente
S01	1	João	Londrina

S03	2	Pedro	Curitiba
S03	3	Maria	Campinas
S02	4	Joana	Fortaleza
S01	5	Marco	Nova York

Fonte: o autor (2021)

Existem outras formas de junções para a fragmentação, possibilitando assim um formato híbrido entre fragmentação horizontal e fragmentação vertical.



VIDEOAULA 03

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda sobre as Desvantagens da Arquitetura Híbrida, assim como a Fragmentação de dados.



INDICAÇÃO DE LEITURA

Leia as 12 primeiras páginas do livro *“From Pots and Vats to Programs and Apps: How Software Learned to Package Itself*, por Gordan Haff e William Henry” para conhecer mais sobre o assunto de Arquitetura em banco de dados.

CURIOSIDADES

No blog *Altistech*, você encontra de forma resumida alguns exemplos de banco de dados para armazenamento. *Link* para acesso abaixo:

Disponível em: <https://www.altistech.com.br/blog-altistech/162-4-armazenamento-de-dados-quais-bancos-utilizar>. Acesso em: 6 jul. 2021.

Encerramento

Tendo como base introdutória para realizarmos consultas utilizando a linguagem SQL, assim como o entendimento de como os dados são interligados até o fornecimento das informações na tela do usuário, destaco os temas abaixo:

1. As consultas SQL facilitam a extração de dados, assim como toda gestão relacionada para alterações no banco de dados, devem ser previamente mapeadas;
2. Existem alguns formatos relacionados a arquiteturas de banco de dados, assim como estratégias para mitigação de possíveis *gaps* de dados.

Agora que você completou esta unidade, você deve ser capaz de:

- Realizar consultas SQL utilizando *Select; Insert; Update e Delete;*
- Utilizar os operadores lógicos para aperfeiçoamento das consultas;
- Discutir a respeito das diversas camadas relacionadas ao banco de dados;
- Aprofundar em temas relacionados à fragmentação de dados.



ENCERRAMENTO

Utilize o QRcode para assistir!



Esperamos que este guia o tenha ajudado compreender a organização e o funcionamento de seu curso. Outras questões importantes relacionadas ao curso serão disponibilizadas pela coordenação.

Grande abraço e sucesso!

