CENTRO UNIVERSITÁRIO SENAC

Disciplina : Banco de Dados

Prof.Me Rômulo Maia

Email :
romulo.fsmaia@sp.senac.br

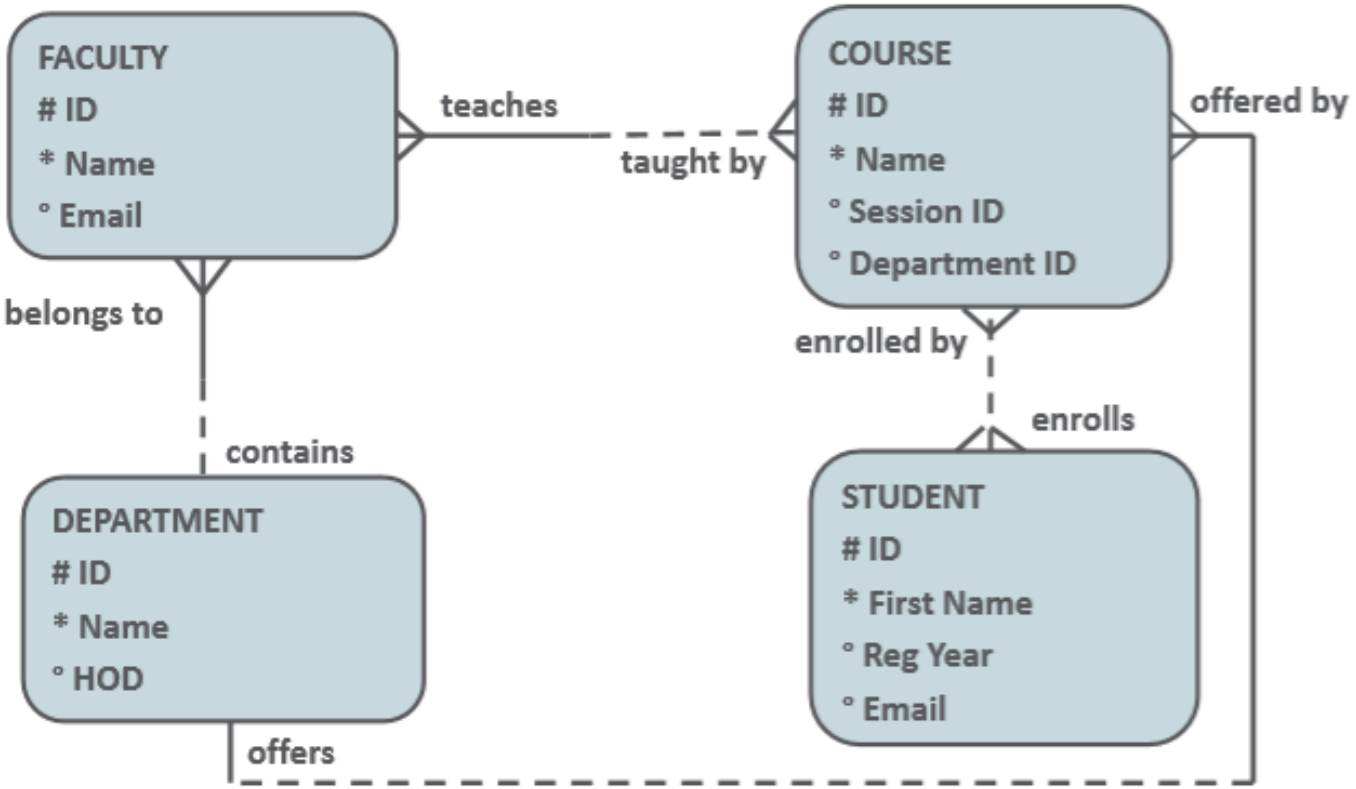
Modelagem Lógica

O modelo de dados lógico:

- Descreve os dados o mais detalhadamente possível, sem se preocupar com sua implementação física no banco de dados.
- É normalmente derivado de um modelo de dados conceitual.
- Inclui todas as entidades, atributos, UIDs e relacionamentos, bem como a opcionalidade e a cardinalidade desses itens.

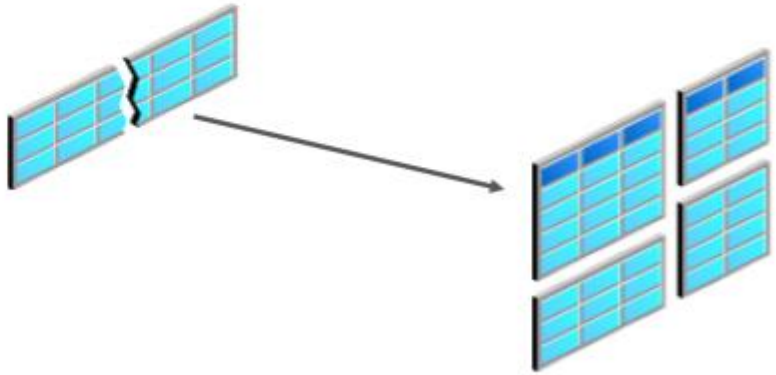
- O modelo lógico é ilustrado com um ERD.

Modelagem Lógica: Exemplo



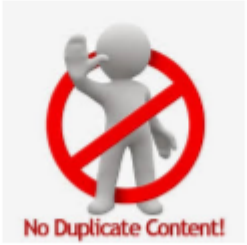
Normalização

- É o processo de organizar os atributos e as tabelas de um banco de dados relacional para minimizar a redundância.
- Ajuda a lidar com anomalias de inserção, atualização e exclusão, garantindo um melhor desempenho do banco de dados.



Por que Você Deve Normalizar Dados?

- Reduzir os dados redundantes no design existente
- Aumentar a integridade dos dados e a estabilidade do design
- Eliminar outros tipos de inconsistências de dados e anormalidades
- Identificar tabelas, colunas e restrições ausentes



O que é Normalização?

- Normalização é um conceito de banco de dados relacional, mas seus princípios se aplicam à modelagem de dados.
- O objetivo é normalizar dados para 3NF antes de transformar o modelo no design relacional.

Regra	Descrição
Primeira Forma Normal (1NF)	Todos os atributos devem ter um único valor.
Segunda Forma Normal (2NF)	Um atributo deve ser dependente do UID inteiro de sua entidade.
Terceira Forma Normal (3NF)	Alguns atributos que não sejam UID podem ser dependentes de outro atributo que não seja UID.

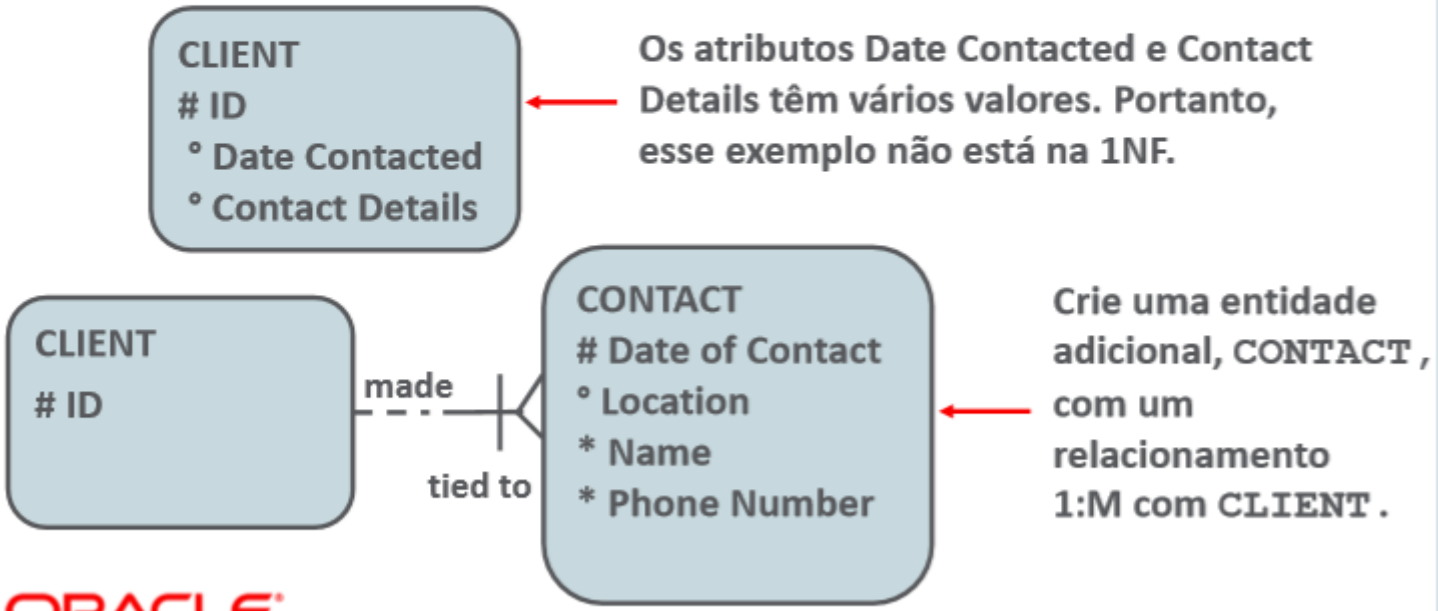
O Que É Primeira Forma Normal? (1NF)

- A Primeira Forma Normal exige que não existam atributos com vários valores.
- Para verificar a 1NF, confirme se cada atributo tem um só valor para cada instância da entidade.
- Se um atributo tiver vários valores, crie uma entidade adicional e relacione-a à entidade original com um relacionamento 1:M.

https://www.youtube.com/watch?v=3kJKJNKiaD4&t=60s

Primeira Forma Normal (1NF)

- Cada atributo deve ter um único valor para cada ocorrência da entidade.



O Que É Segunda Forma Normal? (2NF)

- A Segunda Forma Normal (2NF) requer que um atributo diferente de UID seja dependente (seja uma propriedade ou uma característica) de todo o UID.
- Se o UID for composto, cada atributo deverá ser dependente de todas as partes do UID composto.
- Se um atributo não for dependente de todo o UID, crie uma entidade adicional com o UID parcial.

https://www.youtube.com/watch?v=mHoZZUYVFzk

Segunda Forma Normal (2NF)

- Um atributo deve ser dependente do UID inteiro de sua entidade.

O atributo Bank Location é dependente de `BANK`, não de `ACCOUNT`. Portanto, esse exemplo não está na 2NF. Mova o atributo para a entidade `BANK`.

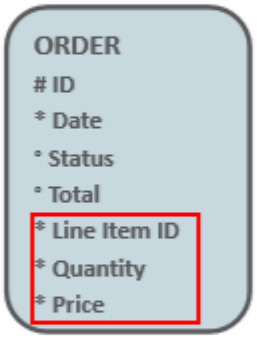
O Que É Terceira Forma Normal? (3NF)

- A regra da Terceira Forma Normal (3NF) determina que nenhum atributo que não seja UID pode depender de outro que não seja UID.
- A Terceira Forma Normal proíbe dependências transitivas.
- Uma dependência transitiva existe quando algum atributo em uma entidade depende de outro atributo que não seja UID nessa entidade.
- Você precisa mover para uma nova entidade todos os atributos que não sejam UID dependentes de outro atributo que não seja UID.

https://www.youtube.com/watch?v=EZvrGEpyNbs

Terceira Forma Normal (3NF)

- Cada atributo depende apenas do UID de sua entidade.
- Mova para uma nova entidade todos os atributos que não sejam UID dependentes de outro atributo que não seja UID.



Os atributos Quantity e Price são dependentes do ID (UID) da tabela ORDER e de Line Item ID (não UID). Portanto, esse exemplo não está na 3NF.

Crie uma nova entidade ORDER ITEM. Mova os atributos Line Item ID, Quantity e Price para a nova entidade e, em seguida, crie um relacionamento identificador.

Tipos de Comandos SQL

- DDL (Data Definition Language) – define estruturas de banco de dados
- DML (Data Manipulation Language) – manipula dados (INSERT, UPDATE, DELETE)
- DQL (Data Query Language) – seleciona (SELECT) dados
- DCL (Data Control Language) – controla o acesso do usuário
- TCL (Transactional Control Language) – gerencia transações de banco de dados

Conjuntos de Comandos da Linguagem SQL

- Linguagem de definição de dados (DDL - Data Definition Language): comandos para criação e manutenção de objetos do banco de dados: CREATE, ALTER, DROP, RENAME e TRUNCATE
- Linguagem para controle de transações: COMMIT, ROLLBACK e SAVEPOINT
- Linguagem para controle de acesso a dados: GRANT e REVOKE

Criação e Destruição de Tabelas

- Formato :

CREATE DATABASE <nome_banco>

USE <nome_banco>

CREATE TABLE <nome_tabela>
(<descrição das colunas>,
<descrição das chaves>);

Shell => mysql –u root –p
Show tables
Show databases;
Desc tabela

http://g2pc1.bu.edu/~qzpeng/manual/MySQL%20Commands.htm

Tipos de Dados

Numéricos

Tipo	Bytes	Mínimo e Máximo
tinyint	1	0 a 255
smallint	2	-32,768 a 32,767
int	4	-2^{31} a $2^{31}-1$
bigint	8	-2^{63} to $2^{63}-1$
decimal(p,s) numeric(p,s)	5 a 17	$-10^{38}+1$ to $10^{38}-1$
smallmoney	4	-214,748.3648 a 214,748.3647
money	8	-922,337,203,685,477.5808 a 922,337,203,685,477.5807
real	4	-3.4^{38} a -1.18^{38}, 0, e 1.18^{38} a 3.4^{38}
float(n)	4 ou 8	-1.79^{308} a -2.23^{308}, 0, e 2.23^{308} a 1.79^{308}

Tipos caracteres

Tipo	Bytes
char(n)	1 byte por caractere, até o máximo de 8000 bytes
varchar(n)	1 byte por caractere armazenado, até o máximo de 8000 bytes
text	1 byte por caractere armazenado, até o máximo de 2GB
nchar(n)	2 bytes por caractere, até o máximo de 4000 bytes
nvarchar(n)	2 bytes por caractere armazenado, até o máximo de 4000 bytes
ntext	2 bytes por caractere armazenado, até o máximo de 2GB

Tipos de data e Hora

Tipo	Faixa de Valores	Precisão	Bytes
smalldatetime	01/01/1900 à 06/06/2079	1 minuto	4
datetime	01/01/1753 à 31/12/9999	0.00333 segundos	8
datetime2	01/01/0001 à 31/12/9999	100 nanosegundos	6 a 8
datetimeoffset	01/01/0001 à 31/12/9999	100 nanosegundos	8 a 10
date	01/01/0001 à 31/12/9999	1 dia	3
time	00:00:00.0000000 a 23:59:59.9999999	100 nanosegundos	3 a 5

Tipos de dados binários

Tipo	Faixa de Valores	Bytes
bit	Null, 0, e 1	1 bit
binary	fixo com dados binários	até 8000
varbinary	dados binários de tamanho variável	até 8000
image	dados binários de tamanho variável	até 2GB

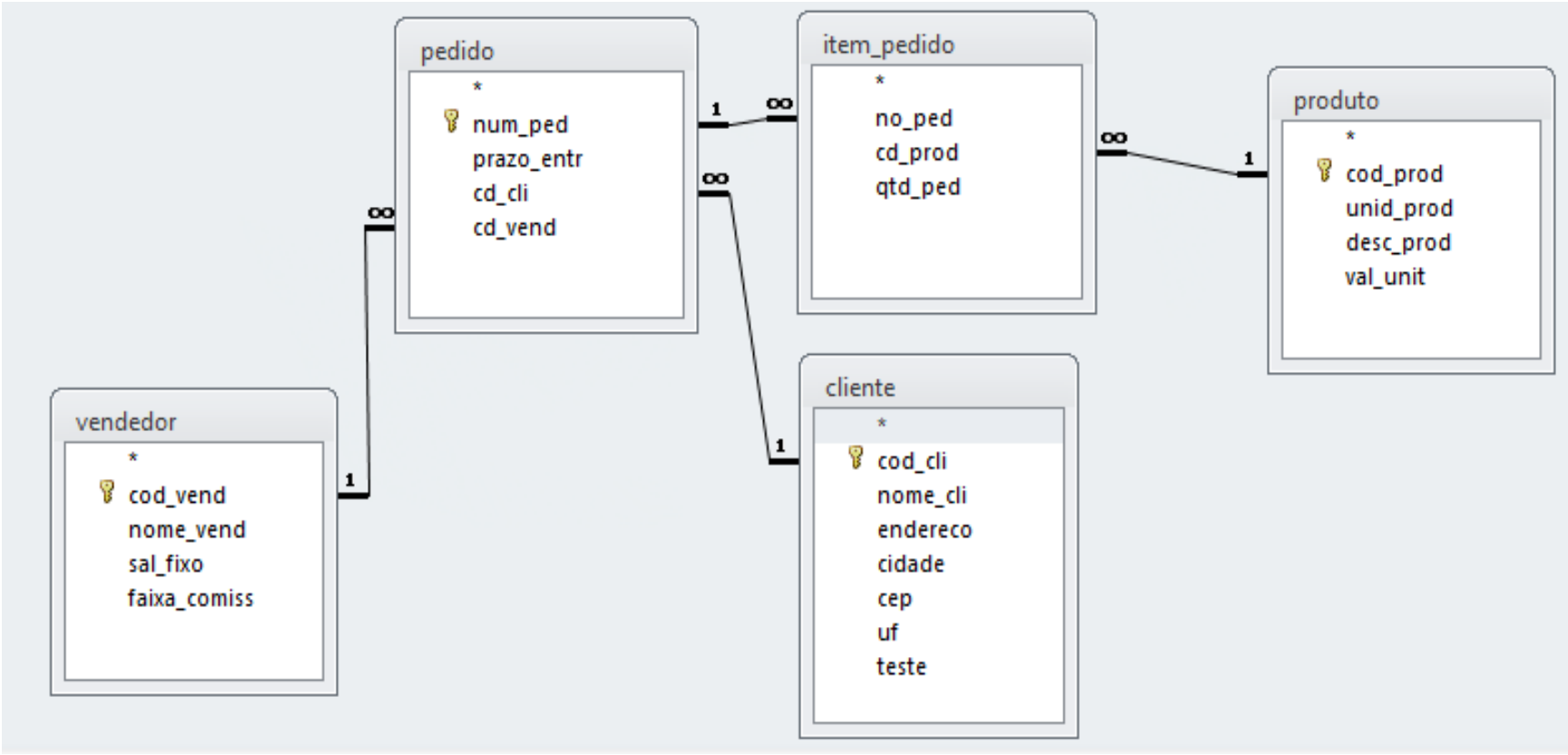
Tipos de Dados

Tipo de Dados	Descrição
VARCHAR2(size)	Dados de caracteres de comprimento variável (É necessário especificar um tamanho máximo, o tamanho mínimo é 1.) Tamanho máximo: 32767 bytes se MAX_SQL_STRING_SIZE = EXTENDED 4000 bytes se MAX_SQL_STRING_SIZE = LEGACY
CHAR(size)	Dados de caracteres de comprimento fixo em bytes. (O tamanho padrão e mínimo é 1; o tamanho máximo é 2.000)
NUMBER(p, s)	Dados numéricos de comprimento variável. A precisão é p e a escala é s. (A precisão é o número total de dígitos decimais, e a escala é o número de dígitos à direita da casa decimal; a precisão pode variar de 1 a 38, e a escala pode variar de -84 a 127.)
DATE	Valores de data e hora até o segundo mais próximo entre 1º de janeiro de 4712 a.C e 31 de dezembro de 9999 d. C.
LONG	Dados de caracteres de comprimento variável (até 2 GB)

Tipos de Dados

Tipo de Dados	Descrição
CLOB	Um objeto de caracteres grande (CLOB) que contém caracteres de um ou de vários bytes. O tamanho máximo é (4 GB - 1) * (DB_BLOCK_SIZE); armazena dados do conjunto de caracteres nacionais.
NCLOB	CLOB que contém caracteres Unicode. Tanto conjuntos de caracteres de largura variável como de largura fixa são suportados e ambos usam o conjunto de caracteres nacionais de banco de dados. O tamanho máximo é (4 GB - 1) * (tamanho do bloco de banco de dados); armazena dados do conjunto de caracteres nacionais.
RAW (Tamanho)	Dados binários brutos de *tamanho* em bytes. É necessário especificar o tamanho para um valor RAW. *Tamanho* máximo: 32767 bytes se MAX_SQL_STRING_SIZE = EXTENDED 4000 bytes se MAX_SQL_STRING_SIZE = LEGACY
LONG RAW	Dados binários brutos de comprimento variável de até 2 GB.
BLOB	Objeto binário grande. O tamanho máximo é (4 GB - 1) * (parâmetro de inicialização DB_BLOCK_SIZE (8 TB a 128 TB)).
BFILE	Dados binários armazenados em um arquivo externo (até 4 GB).
ROWID	String de base 64 que representa o endereço exclusivo de uma linha em sua tabela. Esse tipo de dados destina-se principalmente a valores retornados pela pseudocoluna ROWID

CENTRO UNIVERSITÁRIO SENAC



Script de Criação das Tabelas dos Exemplos

```
create table cliente
(
 cod_cli smallint not null,
 nome_cli varchar(40) not null,
 endereco varchar(40) null,
cidade varchar(20) null,
 cep char(08) null,
 uf char(02) null,
 primary key (cod_cli));
```

```
create table vendedor
(
 cod_vend smallint not null auto_increment,
 nome_vend varchar(40) not null,
 sal_fixo numeric(9,2) not null,
 faixa_comiss char(01) not null,
 primary key (cod_vend));

create table produto
(
 cod_prod smallint not null auto_increment,
 unid_prod char(03) not null,
 desc_prod varchar(20) not null,
 val_unit numeric(9,2) not null,
 primary key (cod_prod));
```

```
create table pedido (
num_ped int not null auto_increment,
prazo_entr smallint not null,
cd_cli smallint not null,
cd_vend int not null,
primary key (num_ped),
foreign key (cd_cli)  REFERENCES CLIENTE (cod_cli),
foreign key (cd_vend)  REFERENCES VENDEDOR (cod_vend));

create table item_pedido (
 no_ped smallint not null  ,
 cd_prod smallint not null,
 qtd_ped float not null,
  foreign key (no_ped) REFERENCES PEDIDO (num_ped),
  foreign key (cd_prod)  REFERENCES PRODUTO (cod_prod) ;
```

Para excluirmos uma tabela existente devemos usar o comando DROP TABLE.

A sua forma geral é: DROP TABLE <nome_tabela>;

onde : <nome_tabela> dever ser substituído pelo nome da tabela a ser excluída.

Exemplos

```
drop table item_pedido;
drop table pedido;
drop table vendedor;
drop table produto;
drop table cliente;
```

Instrução ALTER TABLE

- Use a instrução ALTER TABLE para adicionar, modificar ou eliminar colunas:

```
ALTER TABLE table
ADD         (column data type [DEFAULT expr]
            [, column data type]...);
```

```
ALTER TABLE table
MODIFY      (column data type [DEFAULT expr]
            [, column data type]...);
```

```
ALTER TABLE table
DROP (column [, column] …);
```


No Mysql : Alter table <nome_tabela> modify column <.....> ;
alter table aluno change tefefone telefone varchar(14);

Inserindo, Modificando e Excluindo Registros

1 – Inserindo Registros em uma Tabela

INSERT INTO <nome_tabela> (<lista_de_colunas>) VALUES (<lista_de_valores>);

Problema :

Inserir o produto Tinta de PVC na tabela de produtos.

EX1 : insert into produto values (600,'l','Tinta PVC',15.80);

EX2 : insert into produto (cod_prod,desc_prod) values (200,'pincel');

➔>> exemplo 2 : Cannot insert the value NULL into column 'unid_prod', table 'aulabd.dbo.produto'; column does not allow nulls. INSERT fails.

2 – Modificar Registros em uma Tabela

UPDATE <nome_tabela>
SET {<nome_coluna> = <expressão>}
WHERE <condição_de_seleção>;

Problema 1:

Alterar o preço unitário do Cimento para R$ 5,00.

update produto set val_unit=5.00 where desc_prod='Cimento';

Problema 2 :

Atualizar o salário fixo de todos os vendedores em 27% mais uma bonificação de R$ 100,00.

```
update vendedor
 set sal_fixo=(sal_fixo*1.27+100.00);
```

3 – Exluir em uma Tabela

DELETE FROM <nome_tabela> WHERE <condição_de_seleção>;

Problema:

Excluir todos o itens de pedido que tenham quantidade pedida inferior a 200.

delete from item_pedido where qtd_ped < 200;

Instrução TRUNCATE

- Remove todas as linhas de uma tabela, deixando-a vazia e a sua estrutura intacta.
- É uma instrução DDL, e não uma instrução DML; não é possível desfazê-la facilmente

Sintaxe:

```
TRUNCATE TABLE table_name;
```

Exemplo:

```
TRUNCATE TABLE copy_employees;
```

** Consulte as observações sobre o truncamento das tabelas pai