6. PROJETADO DE BANCO DE DADOS - MODELO FÍSICO

Um **modelo físico** consiste na definição da **abordagem de banco de dados** a ser utilizada, na definição do **Sistema Gerenciador de Banco de Dados** e na escrita das estruturas de dados de acordo com o Sistema Gerenciador de Banco de Dados escolhido.

6.1. TIPOS DE ATRIBUTOS DOS SGBDS ORACLE E MYSQL

SGBD Oracle

Tipo de dado	Descrição
NUMBER(*p*,*s*)	Valor numérico real ou inteiro que possui um número máximo de 38 dígitos, representado por *p*. O *s* representa a quantidade de casas decimais, já incluso nos 38 dígitos. Para valores inteiros não especificar casas decimais.
VARCHAR	Valor alfanumérico com comprimento variável de até 2000 caracteres. Grava somente o digitado. Não usado mais em versões posteriores a 9i.
VARCHAR2	Valor alfanumérico com comprimento variável de até 4000 caracteres. Grava somente o digitado.
DATE	Valor de data e hora entre 1 de janeiro, 4712 A.C. e 31 de dezembro, 9999 D.C. SYSDATE é a data atual do sistema. Formato DD-MM-YYYY
CHAR	Valor alfanumérico de comprimento fixo de até 255 caracteres. Grava todo o tamanho.
LONG	Valor alfanumérico de comprimento máximo de 2 gigabytes. O tamanho não pode ser informado.
CLOB	Valor alfanumérico de comprimento máximo de 4 gigabytes. O tamanho não pode ser informado.

SGBD MySQL

Tipo de dado	Descrição
TINYINT	Valor numérico inteiro de -128 a 127.
SMALLINT	Valor numérico inteiro de -32768 a 32767.
MEDIUMINT	Valor numérico inteiro de -8388608 a 8388607.
INT ***INTEGER***	Valor numérico inteiro de -2147483648 a 2147483647.
BIGINT	Valor numérico inteiro de -9223372036854775808 a 9223372036854775807
FLOAT	Valor numérico real de -3.402823466E+38 a -

	1.175494351E-38, 0, e 1.175494351E-38 a 3.402823466E+38
DOUBLE REAL	Valor numérico real de -1.7976931348623157E+308 a -2.2250738585072014E-308, 0, e 2.2250738585072014E-308 a 1.7976931348623157E+308
DECIMAL	Valor numérico real de -999.99 a 999.99
NUMERIC	Valor numérico real de -9999.99 a 9999.99
DATE	De '1000-01-01' a '9999-12-31'. Formato YYYY-MM-DD
DATETIME	De '1000-01-01 00:00:00' a '9999-12-31 23:59:59'. Formato YYYY-MM-DD HH:MM:SS
TIMESTAMP	De '1970-01-01 00:00:00' válido até 2037. Formato YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD, ou YYMMDD
TIME	De '-838:59:59' a '838:59:59'. Formato HH:MM:SS
YEAR	De 1901 a 2155, e 0000. Formato YYYY
CHAR	Valor alfanumérico de comprimento máximo de até 255 caracteres.
VARCHAR	Valor alfanumérico de comprimento variável de até 65535 caracteres.
TINYTEXT	Valor alfanumérico de comprimento máximo de até 255 caracteres.
TEXT	Valor alfanumérico de comprimento máximo de até 65535 caracteres.
MEDIUMTEXT	Valor alfanumérico de comprimento máximo de até 16777215 caracteres.
LONGTEXT	Valor alfanumérico de comprimento máximo de até 4294967295 caracteres.

6.2. CONEXÃO AO SERVIDOR DE BANCO DE DADOS ORACLE

Configuração da Rede do Laboratório de Informática conectando as salas de laboratório ao Servidor Oracle, banco de dados ORCL (nome de serviço).

Para estabelecer conexão ao Oracle, o servidor deve estar com o Serviço INICIADO. Verifique: Painel de Controle, Ferramentas Administrativas, Serviços: OracleTNSListener e OracleServiceNOMEBD, esses dois serviços devem estar iniciados, conforme ilustrado na Figura 1.

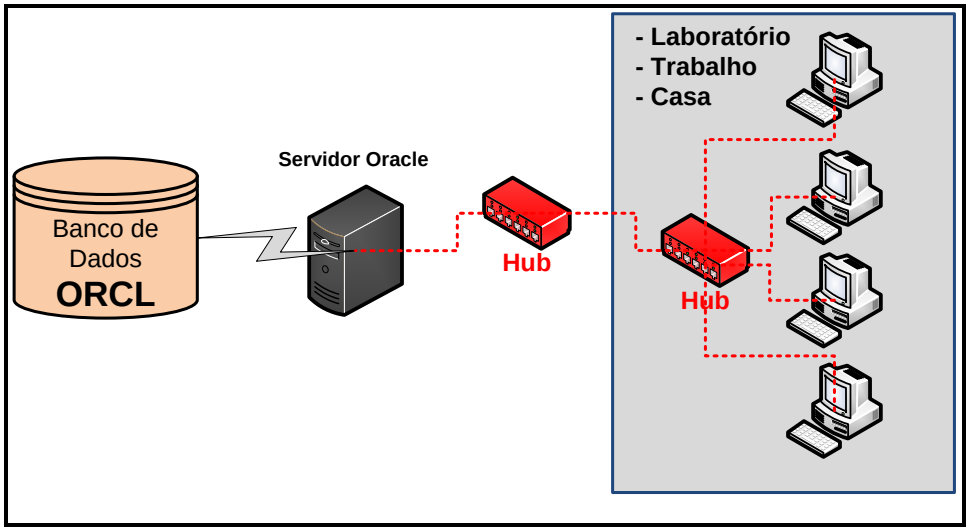

Figura 1. Conexão ao Servidor Oracle (Nome de Serviço **ORCL)**

Explanação prática sobre o uso da Ferramenta **Oracle SQL Developer** para criar conexão com o banco de dados. Entrar com o nome de **Conexão** (pode ser seu próprio nome), **Usuário** (nome criado pelo professor para cada aluno) e **Senha**. O **SID** (nome de serviço do banco de dados) é **ORCL**, a **porta** é **1521**, e o **Host** é o IP **189.56.45.68**, conforme ilustrado na Figura 2.

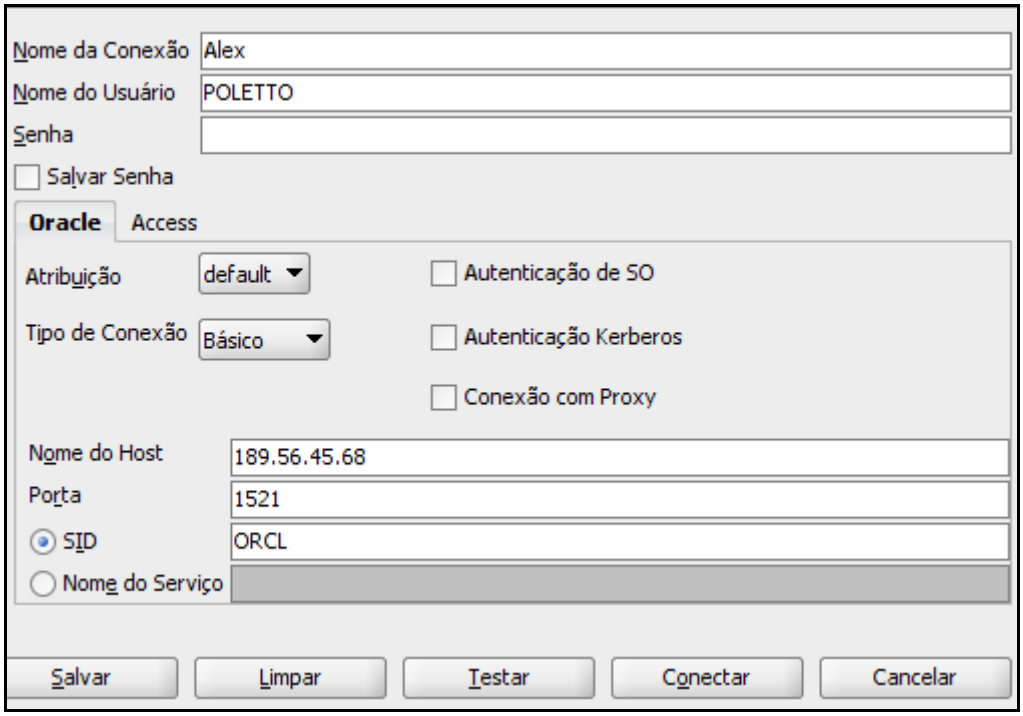

Figura 2. Tela para conexão ao Servidor de Banco de Dados Oracle

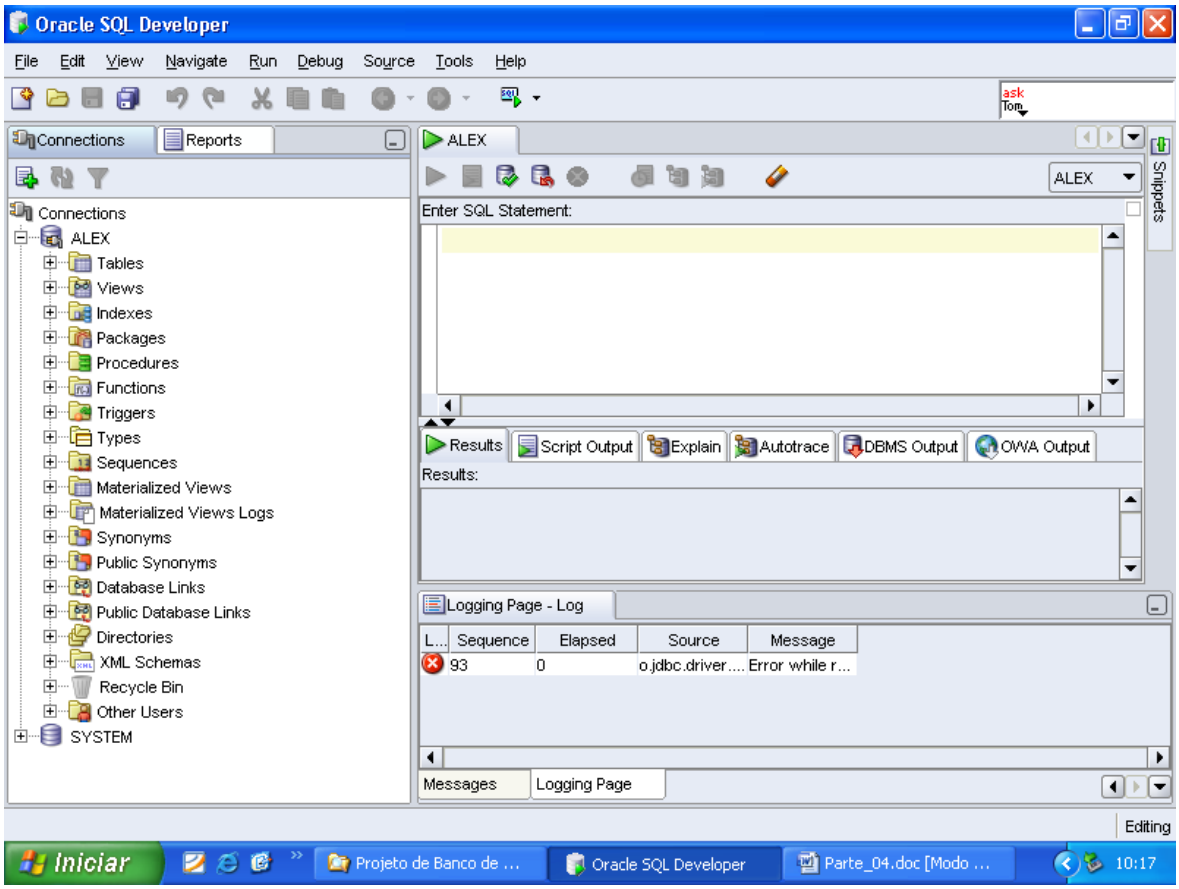

Figura 3. Tela de trabalho junto ao Servidor de Banco de Dados Oracle

6.3. LINGUAGEM SQL-DDL

Embora a SQL seja referida como “linguagem de consulta”, ela pode fazer muito mais do que simplesmente consultar um Banco de Dados. Ela pode definir a estrutura dos dados, bem como modificar essa estrutura de dados, criar restrições para os dados e especificar restrições de segurança.

A SQL se tornou o padrão para linguagens de banco de dados relacionais. Existem diversas versões de SQL. A versão original foi desenvolvida pela IBM no Laboratório de Pesquisa de San José. Essa linguagem, originalmente chamada de Sequel, foi desenvolvida no início dos anos 70. Desde então, a linguagem Sequel foi evoluindo e passou a ser chamada de SQL, isto é, *Structured Query Language* (Linguagem de Consulta Estruturada). A grande vantagem de se usar a SQL, é que inúmeros produtos dão suporte atualmente ao uso da SQL, estabelecendo-se como a linguagem padrão de Bancos de Dados Relacionais.

Nas décadas de 80 e 90, o American National Standards Institute (ANSI), a Internacional Standards Organization (ISO) e a IBM, publicaram os padrões para a SQL: SQL-86; SQL-89 (padrão estendido); SQL-92; SQL:1999.

A SQL:2003 foi uma das últimas versões, sendo publicada como um Padrão Internacional em substituição a SQL:1999. SQL:2003 é popularmente usada por grande parte dos desenvolvedores como um padrão, exceto os que já necessitam usar XML.

A linguagem SQL possui diversas partes, conforme definidas a seguir:

- **Linguagem de definição de dados – DDL**: definição, exclusão e alteração de esquemas, criação de índices, criação de visões, especificação de regras de integridade.
- **Linguagem de manipulação de dados – DML**: inserção, exclusão, modificação e consulta de linhas no banco de dados.
- **Autorização - comandos DCL**: para especificação de direitos de acesso aos objetos (tabelas, visões, etc.).
- **Controle de transações**: comandos para especificação de iniciação e finalização de transações

Linguagem de Definição de Dados (DDL)

A Linguagem de Definição de Dados (Data Definition Language - DDL), específica, através de um conjunto de definições expressas, um esquema de banco de dados, onde o resultado da compilação de comandos de uma DDL é um conjunto de tabelas que são armazenadas em um arquivo chamado dicionário (ou diretório) de dados.

Pode-se considerar diretório de dados como um arquivo que contém metadados, ou seja, “dados sobre dados”.

A estrutura da armazenagem e os métodos de acesso usados em um sistema de bancos de dados são especificados por um conjunto de definições em um tipo especial de DDL chamada linguagem de armazenagem e definição de dados. O resultado da compilação destas definições é um conjunto de instruções para especificar a implementação de detalhes do esquema de bancos de dados que estão escondidos dos usuários.

Dentre as instruções DDL’s estão: Create Table, Alter Table, Drop Table, Truncate, Rename, etc.

6.3.1. Instrução CREATE TABLE

```
CREATE TABLE tabela
(coluna tipo de dados);
```

- O uso da instrução CREATE TABLE possibilita criar tabelas para armazenar dados.
- Para criar uma tabela o usuário deve ter o privilégio CREATE TABLE e uma área de armazenamento para esses objetos.
- *tabela* : é o nome da tabela.
- *coluna* : é o nome da coluna.
- *tipo de dados* : é o tipo de dados e o comprimento da coluna.

Exemplos de instruções:

```
CREATE TABLE tabfun
  (codigo NUMBER(2),
  nome VARCHAR2(50),
  salario NUMBER(10,2),
  data_admissao DATE,
  sexo CHAR(1));
Table created.
SQL> DESCRIBE tabfun
```

```
CREATE TABLE tabfun_m
 AS
  (SELECT codigo, nome, salario, data_admissao
  FROM tabfun
  WHERE sexo='M');
Table created.
DESCRIBE tabfun_m
```

```
CREATE TABLE cop_tabfun
 AS
  (SELECT * FROM tabfun
  WHERE 1=2);
Table created.
Obs: colocando uma condição inútil será copiada somente a estrutura.
DESCRIBE cop_tabfun
```

Tipos de Restrições

```
CREATE TABLE tabela
                (coluna tipo de dados)
                [column_constraint],
                ...
                [table_constraint] [,...]);
```

- As restrições impõem regras no nível da tabela.
- As restrições (CONSTRAINT) evitam que uma tabela seja excluída se houver dependências.
- *tabela* : é o nome da tabela.
- *coluna* : é o nome da coluna.
- *tipo de dados* : é o tipo de dados e o comprimento da coluna.
- *column_constraint*: é uma restrição de integridade como parte da definição da coluna.
- *table_constraint*: é uma restrição de integridade como parte da definição da tabela.
- As restrições devem ser satisfeitas sempre que ocorrer a inserção, atualização ou exclusão de linhas em uma tabela.

Restrições

Restrição	**Descrição**
NOT NULL	Especifica que esta coluna não pode conter um valor nulo.
UNIQUE	Especifica uma coluna ou combinação de colunas cujos valores devem ser exclusivos para todas as linhas na tabela.
PRIMARY KEY	Identifica exclusivamente cada linha da tabela.
FOREIGN KEY	Estabelece e impõem um relacionamento de chave estrangeira entre a coluna e a coluna da tabela referenciada.
CHECK	Especifica uma condição que deve ser atendida.

```
CREATE TABLE depto
    (codigo NUMBER(3),
    descricao VARCHAR2(30),
    CONSTRAINT depto_codigo_pk PRIMARY KEY(codigo),
   CONSTRAINT depto_descricao_un UNIQUE (descricao),
   CONSTRAINT depto_codigo_ck CHECK(codigo BETWEEN 1 AND 100));
```

```
CREATE TABLE professor
   (codigo NUMBER(4),
    nome VARCHAR2(30),
   ...
    depto NUMBER(3) NOT NULL,
     CONSTRAINT professor_codigo_pk PRIMARY KEY (codigo),
     CONSTRAINT professor_depto_fk FOREIGN KEY (depto)
     REFERENCES depto(codigo));
```

6.3.2. Instrução ALTER TABLE

```
ALTER TABLE tabela
  ADD (coluna tipo de dados]...);
```

```
ALTER TABLE tabela
   MODIFY (coluna tipo de dados]...);
```

```
ALTER TABLE tabela
   DROP COLUMN (coluna);
```

- O uso da instrução ALTER TABLE possibilita adicionar, modificar e eliminar colunas em uma tabela.
- A cláusula ADD adiciona colunas.
- A cláusula MODIFY modifica colunas.
- A cláusula DROP COLUMN elimina colunas.
- Para alterar a estrutura de uma tabela o usuário deve ter o privilégio ALTER TABLE.
- *tabela* : é o nome da tabela.
- *coluna* : é o nome da nova coluna.
- *tipo de dados* : é o tipo de dados e o comprimento da nova coluna.

Exemplos de instruções:

```
ALTER TABLE depto
  ADD (localidade VARCHAR2(25));
Table altered.
DESCRIBE depto
```

```
ALTER TABLE depto
  MODIFY (localidade VARCHAR2(35));
Table altered.
DESCRIBE depto
```

```
ALTER TABLE depto
  DROP COLUMN localidade;
Table altered.
DESCRIBE depto
```

Outros exemplos de instruções:

Remove a restrição do depto (chave estrangeira) da tabela PROFESSOR.

```
ALTER TABLE professor
  DROP CONSTRAINT professor_depto_fk;
```

Remove a restrição de chave primária da tabela DEPTO e elimina a restrição de chave estrangeira associada na coluna PROFESSOR.DEPTO.

```
ALTER TABLE depto
  DROP PRIMARY KEY CASCADE;
```

Desativar restrições de integridade dependentes, usando a cláusula DISABLE e CASCADE

```
ALTER TABLE professor
  DISABLE CONSTRAINT professor_depto_fk CASCADE;
```

Ative restrições de integridade dependentes, usando a cláusula ENABLE

```
ALTER TABLE professor
  ENABLE CONSTRAINT professor_depto_fk;
```

6.3.3. Instrução DROP TABLE

```
DROP TABLE tabela;
```

- O uso da instrução DROP TABLE possibilita eliminar uma tabela por completo, tanto os dados quanto sua estrutura.
- Todas as transações pendentes sofrerão COMMIT.
- Todos os índices serão eliminados.
- Não existe a possibilidade de ROLLBACK, de recuperação da tabela.
- Para eliminar uma tabela o usuário deve ter o privilégio DROP TABLE.
- *tabela* : é o nome da tabela.

Exemplos de instruções:

```
DROP TABLE depto;
Table dropped.
```

6.3.4. Instrução RENAME

```
RENAME old_name  TO new_name;
```

- O uso da instrução RENAME possibilita renomear uma tabela.
- Não existe a possibilidade de ROLLBACK, de recuperação da tabela.
- Para eliminar uma tabela o usuário deve ser o proprietário da tabela.
- *old_name* : é o nome antigo da tabela.
- *new_name* : é o novo nome da tabela.

Exemplos de instruções:

```
RENAME depto TO depart;
Table renamed.
```

6.3.5. Instrução TRUNCATE TABLE

```
TRUNCATE TABLE tabela;
```

- O uso da instrução TRUNCATE TABLE possibilita remover todas as linhas de uma tabela e liberar o espaço de armazenamento usado por esta tabela.
- Não existe a possibilidade de ROLLBACK, de recuperação da remoção das linhas.
- Para truncar uma tabela o usuário deve ser o proprietário da tabela ou ter privilégios de TRUNCATE TABLE.
- *tabela* : é o nome da tabela.

Exemplos de instruções:

```
TRUNCATE TABLE depto;
Table truncated.
```

6.3.6. Consultar METADADOS (Dicionário de Dados)

Tabela de Metadados USER_TABLES: contém as definições de tabelas

```
Para ver a estrutura da tabela de metadados
DESC USER TABLES

Para ver todos os nomes e definições de tabelas
SELECT table_name
FROM user_tables;
```

Tabela de Metadados USER_CONSTRAINTS: contém as definições de restrições

```
SELECT constraint_name, constraint_type, search_condition
FROM user_constraints
WHERE table_name = 'PROFESSOR';
```

Visualize as colunas associadas aos nomes de restrições na view *USER_CONS_COLUMNS*

```
SELECT constraint_name, column_name
FROM user_cons_columns
WHERE table_name = 'PROFESSOR';
```

6.4. DEFINIÇÃO DA ABORDAGEM DE BANCO DE DADOS

Será utilizada a Abordagem Relacional.

6.5. DEFINIÇÃO DO SISTEMA GERENCIADOR DE BANCO DE DADOS

Os exemplos serão criados no sistema gerenciador de banco de dados Oracle, Porém, serão apresentados as instruções para futura criação no sistema gerenciador de banco de dados MySQL.

Para a criação das estruturas de dados no Oracle será utilizada a ferramenta Oracle SQL Developer. Para a criação das estruturas de dados no MySQL pode-se usar a ferramenta Navecat.

6.6. TRANSFORMANDO O MODELO CONCEITUAL EM SQL-DDL

Modelo de Dados Físico com base nos tipos de dados do Oracle

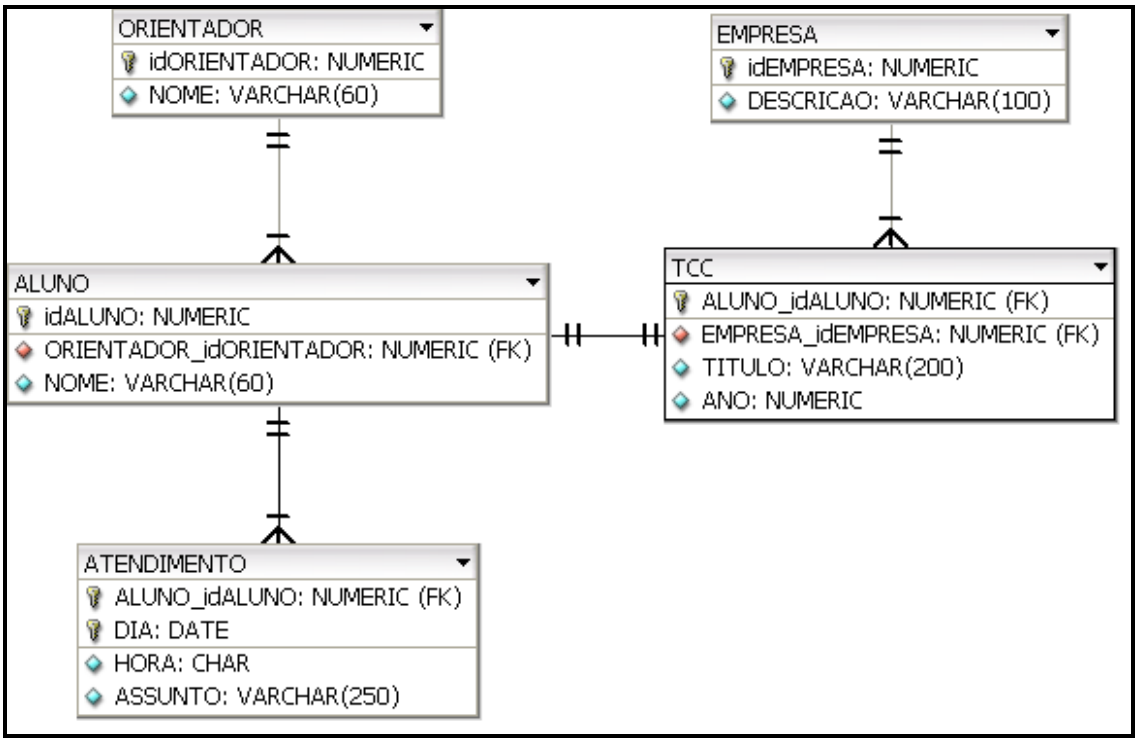

Figura 4. Modelo de Dados do TCC para Oracle

Usar a Ferramenta Oracle SQL Developer

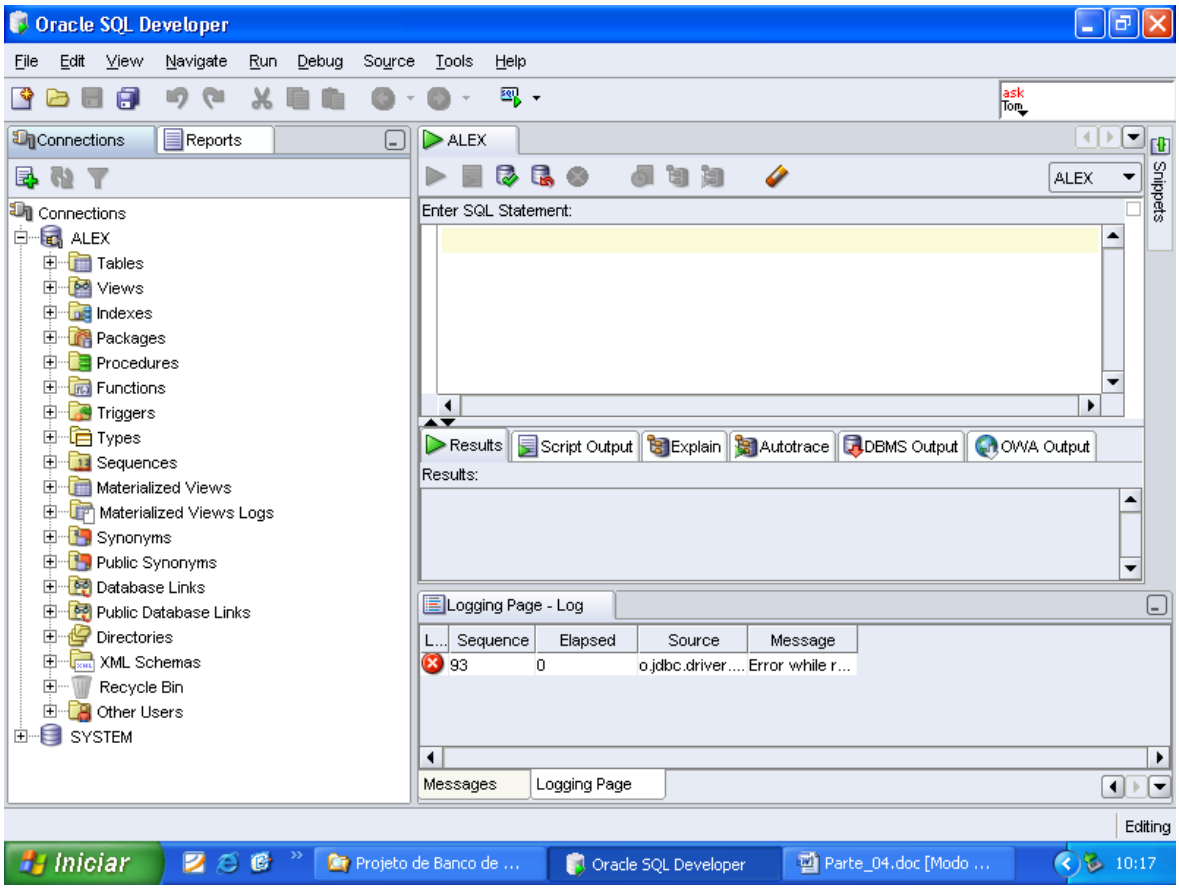

Figura 5. Tela Principal da Ferramenta SQL Developer

Estrutura de dados para o Oracle

```
CREATE TABLE EMPRESA
(idEMPRESA NUMBER(5),
 DESCRICAO VARCHAR(100),
 CONSTRAINT EMPRESA_PK PRIMARY KEY(idEMPRESA));

CREATE TABLE ORIENTADOR
(idORIENTADOR NUMBER(3),
 NOME VARCHAR(60),
 CONSTRAINT ORIENTADOR_PK PRIMARY KEY(idORIENTADOR));

CREATE TABLE ALUNO
(idALUNO NUMBER(10),
 ORIENTADOR_idORIENTADOR NUMBER(3),
 NOME VARCHAR(60),
 CONSTRAINT ALUNO_PK PRIMARY KEY(idALUNO),
 CONSTRAINT ALUNO_ORIENTADOR_FK FOREIGN KEY
```

```
(ORIENTADOR_idORIENTADOR) REFERENCES ORIENTADOR(idORIENTADOR));

CREATE TABLE ATENDIMENTO
(ALUNO_idALUNO NUMBER(10),
 DIA DATE,
 HORA CHAR(5),
 ASSUNTO VARCHAR(250),
 CONSTRAINT ATENDIMENTO_PK PRIMARY KEY(ALUNO_idALUNO, DIA),
 CONSTRAINT ATENDIMENTO_ALUNO_FK FOREIGN KEY(ALUNO_idALUNO)
   REFERENCES ALUNO(idALUNO));

CREATE TABLE TCC
(EMPRESA_idEMPRESA NUMBER(5),
 ALUNO_idALUNO NUMBER(10),
 TITULO VARCHAR(200) NOT NULL,
 ANO NUMBER(4) NOT NULL,
 CONSTRAINT TCC_PK PRIMARY KEY(ALUNO_idALUNO),
 CONSTRAINT TCC_EMPRESA_FK FOREIGN KEY(EMPRESA_idEMPRESA)
   REFERENCES EMPRESA(idEMPRESA),
 CONSTRAINT TCC_ALUNO_FK FOREIGN KEY(ALUNO_idALUNO)
   REFERENCES ALUNO(idALUNO));
```

Figura 6. *Scripts* para a criação das estruturas de dados no Oracle

Modelo de Dados Físico com base nos tipos de dados do MySQL

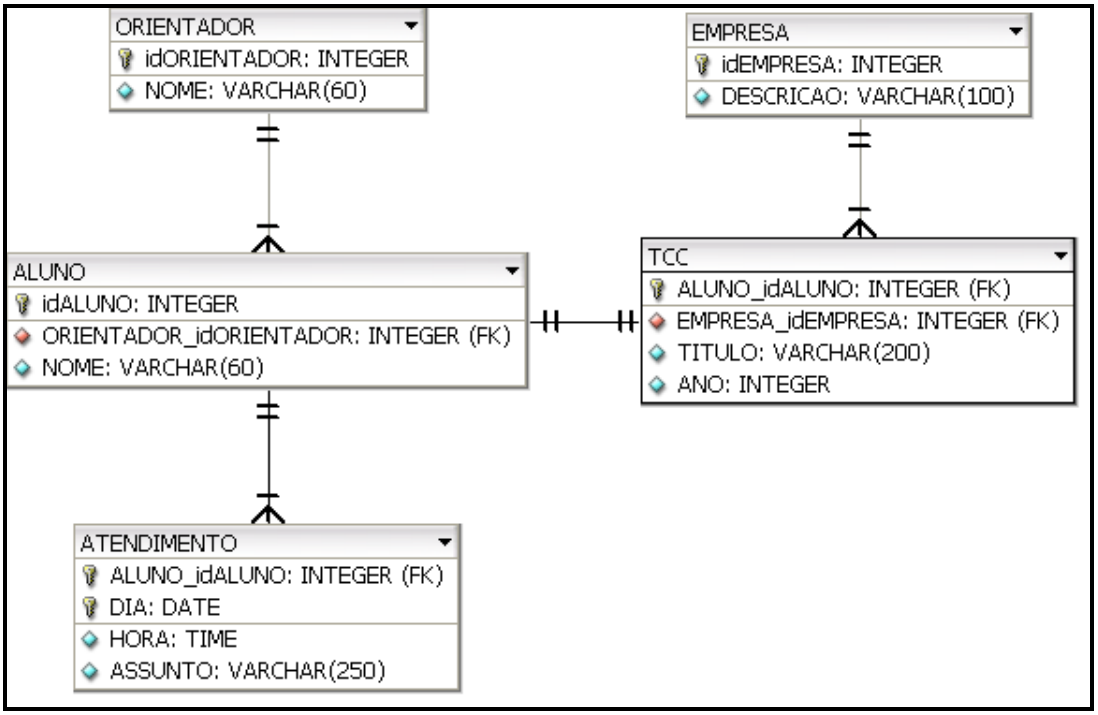

Figura 7. Modelo de Dados do TCC para MySQL

Usar a Ferramenta Navecat

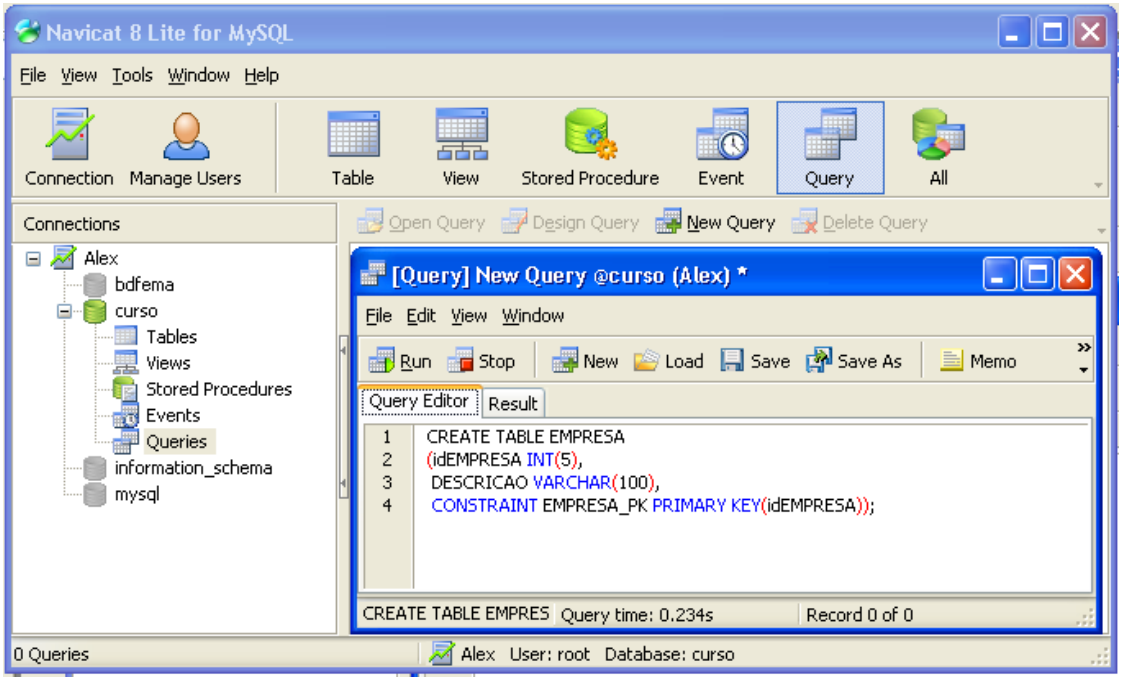

Figura 8. Tela Principal da Ferramenta Navecat

Estrutura de dados para o MySQL

```
CREATE TABLE EMPRESA
(idEMPRESA INT(5),
 DESCRICAO VARCHAR(100),
 CONSTRAINT EMPRESA_PK PRIMARY KEY(idEMPRESA));

CREATE TABLE ORIENTADOR
(idORIENTADOR INT(3),
 NOME VARCHAR(60),
 CONSTRAINT ORIENTADOR_PK PRIMARY KEY(idORIENTADOR));

CREATE TABLE ALUNO
(idALUNO INT(10),
 ORIENTADOR_idORIENTADOR INT(3),
 NOME VARCHAR(60),
 CONSTRAINT ALUNO_PK PRIMARY KEY(idALUNO),
 CONSTRAINT ALUNO_ORIENTADOR_FK FOREIGN KEY
(ORIENTADOR_idORIENTADOR) REFERENCES ORIENTADOR(idORIENTADOR));

CREATE TABLE ATENDIMENTO
(ALUNO_idALUNO INT(10),
 DIA DATE,
 HORA TIME,
 ASSUNTO VARCHAR(250),
```


```
 CONSTRAINT ATENDIMENTO_PK PRIMARY KEY(ALUNO_idALUNO, DIA),
 CONSTRAINT ATENDIMENTO_ALUNO_FK FOREIGN KEY(ALUNO_idALUNO)
   REFERENCES ALUNO(idALUNO));

CREATE TABLE TCC
(EMPRESA_idEMPRESA INT(5),
 ALUNO_idALUNO INT(10),
 TITULO VARCHAR(200) NOT NULL,
 ANO INT(4) NOT NULL,
 CONSTRAINT TCC_PK PRIMARY KEY(ALUNO_idALUNO),
 CONSTRAINT TCC_EMPRESA_FK FOREIGN KEY(EMPRESA_idEMPRESA)
   REFERENCES EMPRESA(idEMPRESA),
 CONSTRAINT TCC_ALUNO_FK FOREIGN KEY(ALUNO_idALUNO)
   REFERENCES ALUNO(idALUNO));
```

Figura 9. *Scripts* para a criação das estruturas de dados no MySQL