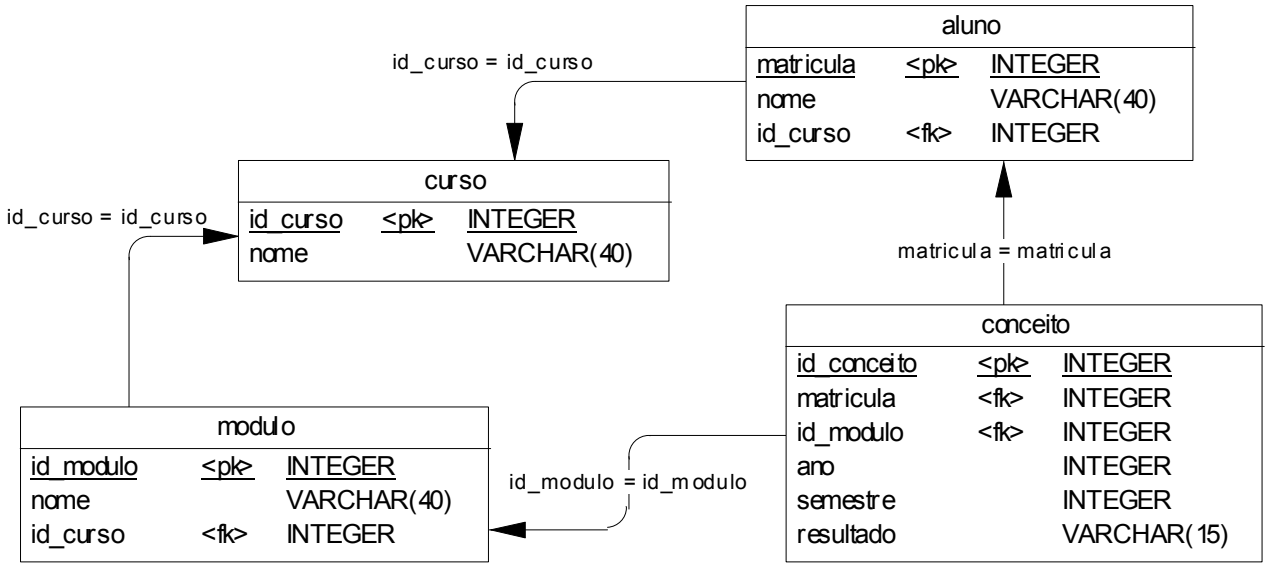
Comandos DDL

Estudo de Caso – Controle Acadêmico Simplificado

Uma escola contém vários cursos, onde cada aluno possui uma matricula num determinado curso. Estes cursos, por sua vez, possuem módulos, aos quais serão atribuídos resultados finais a cada aluno. O DER da figura abaixo representa uma modelagem física, de acordo com o padrão ANSI da linguagem SQL.



MODELAGEM FÍSICA DE EXEMPLO.

Simbologia

Os comandos SQL serão apresentados a seguir segundo uma simbologia padrão:

- As palavras em **negrito** são obrigatórias. Na redação do comando SQL devem aparecer, sempre.
- Os termos entre "<" e ">" devem ser substituídos pela expressão indicada. Por exemplo: O termo <nome-tabela>, num comando SQL, deve ser substituído por uma tabela do BD.
- Os termos entre "[" e "]" indicam opcionalidade.
- Os termos entre "{" e "}" indicam uma escolha obrigatória dentre as opções.

Importante salientar que podem existir variações nos comandos em relação ao SGBD escolhido na modelagem física dos dados.

Comandos DDL

CRIAR TABELA

Sintaxe:
```
create table <tabela>
(
      <coluna> <tipo-do-dado>    [not null] [not null with default] [,
      <coluna> <tipo-do-dado>    [not null] [not null with default] ...]
      primary key (<coluna-pk>)
      foreign key (<coluna-fk>) references <tabela-pai> (<coluna-pk-pai>)
)
```

Semântica: cria uma nova tabela no banco de dados em uso.

<tabela> : define o nome da tabela
<coluna>: define o nome da coluna. A definição das colunas de uma tabela é feita relacionando-as uma após a outra, separadas por vírgula.
<tipo-do-dado> : define o tipo e o tamanho da coluna definida. Os tipos de dados mais comuns são:

- SmallInt: número de 2 bytes, entre–32768 e +32767
- Integer: número de 4 bytes, entre -2147483648 e +2147483647

- Decimal(n,m): número com no máximo 15 dígitos, onde "n" são dígitos inteiros e "m" as casas decimais.
- Varchar(n): *string* de até "n" caracteres ("n", no máximo, igual 254)
- Char(n): *string* de exatos "n" caracteres ("n", no máximo, igual 254)
- Long Varchar: *string* de comprimento maior que 254 caracteres.
- Date: campo no formato de uma data
- Time: campo no formato de um horário

<not null> : define uma coluna de preenchimento obrigatório

<not null with default> : define uma coluna de preenchimento obrigatório e que, no momento da inclusão de uma linha na tabela, tenha um valor pré-definido, tais como:

- número zero, para o tipo de dado numérico
- caracter vazio, para o tipo de dado *string*
- data corrente, para o tipo de dado *date*
- horário corrente, para o tipo de dado *time*

primary key (<coluna-pk>) : define a coluna que será a chave primária da tabela. No caso de mais de uma coluna ser chave primária, separá-las por vírgula.

foreign key (<coluna-fk>) references <tabela-pai> (<coluna-pk-pai>): define a <coluna-fk>, de <tabela>, como chave estrangeira. A <tabela-pai> define a tabela-pai relacionada a *fk* e <coluna-pk-pai> define a coluna *pk*, de <tabela-pai>

Exemplos:

```
create table curso
(
    id_curso     INTEGER              not null,
    nome         VARCHAR(40)          not null,
    primary key (id_curso)
);

create table modulo
(
    id_modulo    INTEGER              not null,
    nome         VARCHAR(40)          not null,
    id_curso     INTEGER              not null,
    primary key (id_modulo),
    foreign key (id_curso) references curso (id_curso)
);
```

CONSTRAINTS: *constraints* são regras agregadas a colunas ou tabelas. Assim, pode-se definir como obrigatório o preenchimento de uma coluna que tenha um valor-padrão quando uma linha for incluída na tabela ou quando aceitar apenas alguns valores predefinidos. No caso de regras aplicadas a tabelas, tem-se a definição de chaves primárias e estrangeiras.

Exemplos:

```
create table aluno
(
   matricula    INTEGER      not null,
   nome         VARCHAR(40)  not null,
   sexo         CHAR(1)      check (upper(sexo) = 'M' or upper(sexo) = 'F'),
   cpf          NUMERIC(11)  unique,
   id_curso     INTEGER      not null,
   primary key (matricula),
   foreign key (id_curso) references curso (id_curso)
);
```

Constraint de chave primária: pode ser escrita de duas formas:

```
primary key (matricula)
```

ou então

```
constraint aluno_pk primary key (matricula)
```

Constraint de chave estrangeira: pode ser escrita de duas formas:

```
foreign key (id_curso) references curso (id_curso)
```

ou então

```
constraint aluno_curso_fk foreign key (id_curso) references curso (id_curso)
```

Opcionalmente, pode-se acrescentar definições das ações que deverão ser executadas em caso de atualização ou remoção de linhas da tabela.

Sintaxe:

```
constraint <nome_da_constraint> foreign key (<coluna-fk>)
        references <tabela-pai> (<coluna-pk-pai)
        [ on update { set null | set default | cascade | no action | restrict } ]
        [ on delete { set null | set default | cascade | no action | restrict } ]
```

Onde:

update: é a operação de atualizar (alterar) alguma informação de uma linha da tabela que contém a referência (a <tabela-pai>)
delete: é a operação de remoção de uma linha da tabela
set null: altera o conteúdo da coluna para nulo, sem deixar valores inconsistentes
set default: altera o conteúdo da coluna para o valor especificado na cláusula *default*, caso houver
cascade: atualiza ou remove todos os registros que se relacionam a eles
no action: em caso de atualização, não modifica os valores que se relacionam a eles
restrict: não permite a exclusão da chave primária

```
Exemplo:  constraint aluno_curso_fk foreign key (id_curso)
              references curso (id_curso)
              on update cascade
              on delete restrict
```

Constraint de valor padrão: `condicao VARCHAR(11) default 'MATRICULADO'`

Constraint de valor não nulo: `nome VARCHAR(40) not null`

Constraint de valor único: `cpf NUMERIC(11) unique`

Constraint de definição de domínio:
```
sexo CHAR(1) check (upper(sexo) = 'M' or
                    upper(sexo) = 'F'),
```

ALTERAR TABELA

Sintaxe:
```
alter table <tabela>
        { [ drop <coluna> ] |
          [ add <coluna> <tipo-do-dado> [not null] [not null with default]] |
          [ rename  <coluna> <novo-nome-coluna>] |
          [ rename  table <novo-nome-tabela>] |
          [ add primary key ( <coluna-pk> ) ] |
          [ drop primary key ( <coluna-pk> ) ] |
          [ add foreign key (<coluna-fk>) references <tabela-pai> (<coluna-pk-pai>)
          [ drop foreign key (<coluna-fk>) references <tabela-pai> ]
         }
```

Semântica: altera definições na estrutura de uma tabela do BD, acrescentando, alterando e removendo nomes, formatos das colunas e regras de integridade referencial.

<tabela> : define o nome da tabela
drop <coluna> : remove a coluna da estrutura da tabela
add <coluna> <tipo-do-dado> [not null] [not null with default] : acrescenta uma nova coluna. No caso da existência de linhas nesta tabela, o valor da nova coluna será de acordo com as definições das cláusulas *not null* e *not null with default*.
rename <coluna> <novo-nome-coluna>: troca o nome da coluna
rename table <tabela>: troca o nome da tabela
add primary key (<coluna-pk>): define uma chave primária à uma nova coluna acrescentada na tabela.
drop primary key (<coluna-pk>): remove a definição de chave primária da coluna
add foreign key (<coluna-fk>) references <tabela-pai> (<coluna-pk-pai>) acrescenta uma nova chave estrangeira.
drop foreign key (<coluna-fk>) references <tabela-pai> : remove a definição de chave estrangeira.

Exemplos:

```
create table aluno
(
    matricula    INTEGER              not null,
    nome         VARCHAR(40)          not null,
    id_curso     INTEGER              not null,
    primary key (matricula)
);

alter table aluno
    add foreign key (id_curso) references curso (id_curso);
```

OBS: Estas duas definições acima são equivalentes a:

```
create table aluno
(
    matricula    INTEGER              not null,
    nome         VARCHAR(40)          not null,
    id_curso     INTEGER              not null,
    primary key (matricula),
    foreign key (id_curso) references curso (id_curso)
);
```

REMOVER TABELA

Sintaxe: **drop table** <tabela>

Semântica: remove uma tabela do banco de dados. Esta operação remove linhas, estrutura e índices de acesso associados à tabela.

<tabela>: define o nome da tabela

Exemplo:

```
drop table curso;
```

CRIAR ÍNDICE

Sintaxe: **create** [**unique**] **index** <índice>
on <tabela> **(** <coluna> [**ASC**] [**DESC**] [, <coluna> [**ASC**] [**DESC**]] **)**

Semântica: cria uma estrutura de índice de acesso para uma determinada coluna de uma tabela. Um índice permite um acesso mais rápido aos dados e pode ser criado a partir de uma ou mais colunas da tabela. Toda chave primária possui um índice definido.

<índice>: define o nome do índice. A cláusula **unique** define que não existirão duas linhas com o mesmo conteúdo de <coluna>. A definição do índice de uma chave primária contém **unique**.
<tabela>: define o nome da tabela que contém a coluna que terá o índice
<coluna>: define a coluna da tabela. As opções ASC/DESC representam, respectivamente, uma ordenação ascendente e descendente.

Exemplos:

```
create unique index curso_pk on curso (id_curso asc);

create unique index modulo_pk on modulo (id_modulo asc);

create unique index aluno_pk on aluno (matricula asc);
create index aluno_nome on aluno (nome asc);
```

REMOVER ÍNDICE

Sintaxe: **drop index** <índice>

Semântica: remove uma estrutura de índice definida numa tabela do BD.

<índice>: define o nome do índice

Exemplo:

```
drop index curso_pk;
drop index aluno_nome
```

A figura abaixo mostra um arquivo de script gerado por um software de modelagem de DER utilizando um SGBD com as instruções SQL padrão ANSI. Este arquivo contém os comandos DDL do SQL que, quando executados, deverão criar as tabelas do banco de dados.

```
create table curso
(
    id_curso     INTEGER              not null,
    nome         VARCHAR(40)          not null,
    primary key (id_curso)
);
create unique index curso_pk on curso (id_curso asc);


create table modulo
(
    id_modulo    INTEGER              not null,
    nome         VARCHAR(40)          not null,
    id_curso     INTEGER              not null,
    primary key (id_modulo)
);
create unique index modulo_pk on modulo (id_modulo asc);


create table aluno
(
    matricula    INTEGER              not null,
    nome         VARCHAR(40)          not null,
    id_curso     INTEGER              not null,
    primary key (matricula)
);
create unique index aluno_pk on aluno (matricula asc);


create table conceito
(
    id_conceito  INTEGER              not null,
    matricula    INTEGER              not null,
    id_modulo    INTEGER              not null,
    ano          INTEGER              not null,
    semestre     INTEGER              not null,
    resultado    VARCHAR(15)                  ,
    primary key (id_conceito)
);
create unique index conceito_pk on conceito (id_conceito asc);


alter table modulo   add foreign key (id_curso)  references curso  (id_curso);
alter table aluno    add foreign key (id_curso)  references curso  (id_curso);
alter table conceito add foreign key (matricula) references aluno  (matricula);
alter table conceito add foreign key (id_modulo) references modulo (id_modulo);
```

ARQUIVO DE SCRIPT PARA A CRIAÇÃO DAS TABELAS DO SGBD