

Base de Dados

Passos Básicos para o design de uma BDR

1. Determinar o propósito do sistema;
2. Determinar quais as entidades e atributos;
3. Identificar chaves primárias;
4. Determinar relações entre tabelas;
5. Redefinir o design (normalização).

→ Os atributos não podem ser do tipo composto ou multi - valor (')

→ Cada tabela tem de ter uma chave primária.

→ Podem existir vários atributos cujos valores identificam exclusivamente uma ocorrência dessa tabela : chaves candidatas. A chave primária é uma das chaves candidatas.

→ Uma chave primária pode ser formada pela combinação de pelo menos dois ou mais atributos sendo nesses casos chamada 'chave composta'.

Ex:

Nota
nr_matriula Pk
codigo_disciplina Pk
nota_final_disciplina

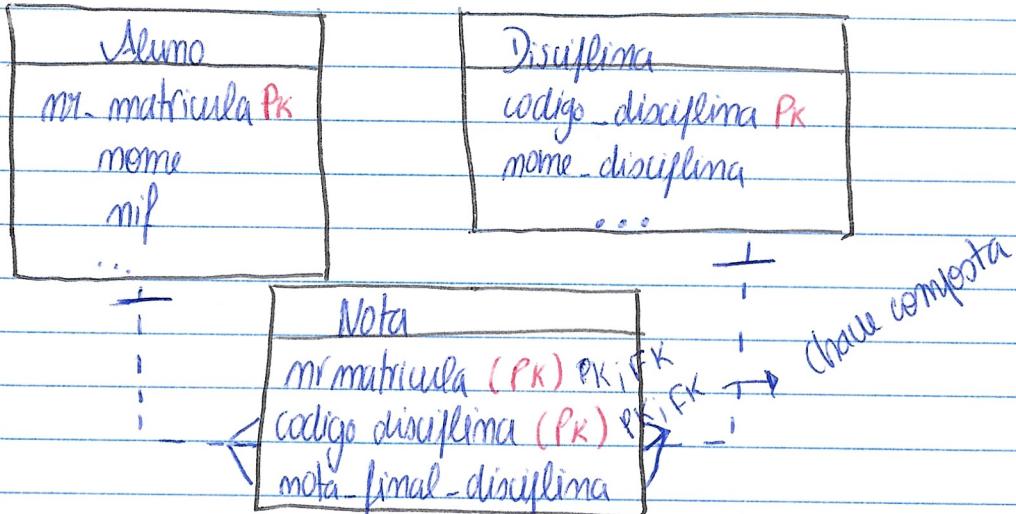
→ A chave primária também é usada para fazer referência a outras tabelas → aparece o conceito de chave estrangeira;

→ No modelo relacional não deve haver o tipo de cardinalidade de muitos - para - muitos

* Um - para - muitos - A chave primária de um lado forma - se numa chave estrangeira no lado oposto (colocar sempre onde está a cardinalidade N).

* Um - para - um - chave primária no lado obrigatório forma - se uma chave estrangeira no lado optional.

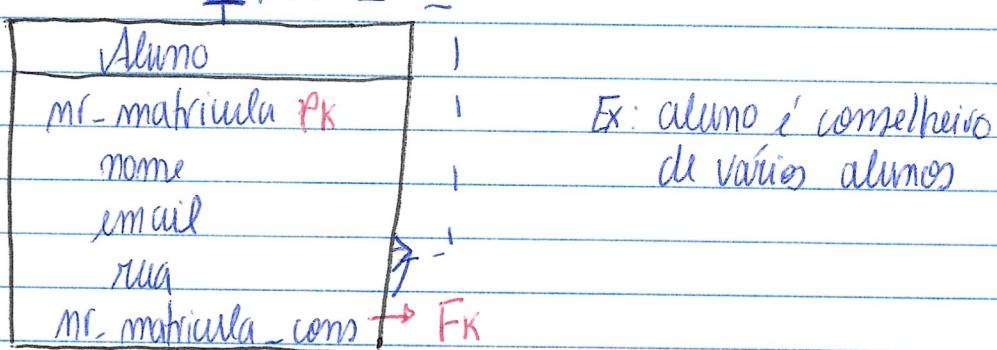
* Muitos - para - muitos - Criar uma nova relação com as chaves primárias das duas entidades como sua chave primária.



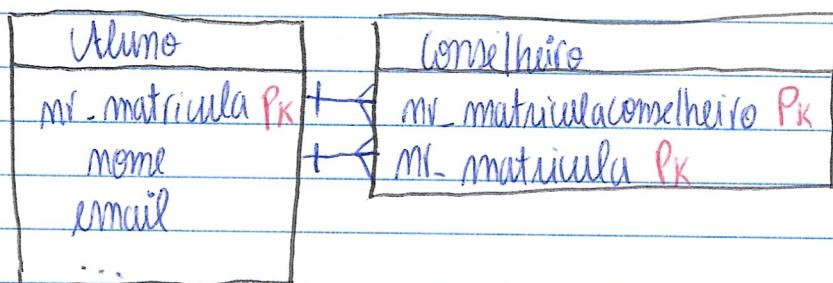
→ No modelo relacional não pode haver relacionamentos de muitos - para - muitos.

Relações unárias :

> Um para muitos
chave estrangeira recursiva na mesma relação;



> Muitos - para - muitos



Normalização

Restrições de integridade

→ de entidade
→ referencial
→ domínio
→ semântica

Integridade de entidade

↳ Nenhum atributo pertencente a uma chave primária poderá ter um valor nulo.

Integridade referencial

↳ Quando há valores na chave estrangeira, esses valores têm de existir na chave primária

Integridade de domínio

Domínio: conjunto de valores que um atributo pode assumir

↳ O valor de um atributo de uma entidade está contido no domínio desse atributo, nessa entidade

Ex: Nome: VARCHAR(20) - Alexandra Maria Oliveira



viola a regra porque tem + 20 caracteres;

Integridade colunas: ... note o uso do CHECK(nota BETWEEN 0 and 20)



equivalente a um if na programação

Integridade Semântica

Ex: O salário de um empregado deve ser menor ou igual ao do seu supervisor

→ Não se deve vender produtos a um cliente que deu mais do que x €.

→ Restrições garantidas através da implementação de TRIGGERS → implica programação



Processo de normalização

Evita redundâncias desnecessárias;

Evita problemas associados à inserção, eliminação e atualização de dados;

Dependências funcionais

1^a

Forma [Número de aluno] → [Nome de aluno]

Normal

↳ nome de aluno depende funcionalmente do número de aluno, ou, número de aluno determina o nome do aluno.

Dependência funcional total: Número de aluno, cod. disciplina → Nota

↳ quando a chave

primária é composta
(PK; FK)

① Definir chave primária de todas tabelas.

→ todos os atributos dependem funcionalmente da chave primária.

2^a

Forma → Cada atributo não chave depende funcionalmente da totalidade da chave.

→ Se a relação só tem um atributo como chave primária e se essa relação já estiver na 1FN, então a relação também se encontra na 2FN.

3^a

Forma → Esquecer a chave primária e ver se há algum atributo que depende de qualquer um dos restantes

Dependência funcional transitiva

Línguagem SQL

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    ...  
) ;
```

Tipos de dados

Char(n) cadeia de caracteres de tamanho fixo (n)

Varchar(n) : // com tamanho máximo (n)

int: números inteiros (4 bytes)

Number(precisão, escala): números reais sem limite tamanho

date: data entre 4712-01-01 AC e 4712-12-31 DC

timestamp: Data + hora no mesmo campo

number (5,2)

Ex: 123,45 ou 1,23

↓
pode ter
até 5 dígitos
no total

→ pode ter até
2 casas
decimais

Valores por omissão (default)

Ex: create table aluno (

bi integer

media float number (4,2) default 0

↳ se nenhum valor for especificado
durante a inserção do registo, o
valor padrão será 0.

Restrições de integridade - CHECK

Ex: create table aluno (
mediaCurso number(4,2) default 0, constraint ck_media check (mediaCurso >= 0 and mediaCurso <= 10),
nome varchar(50) not null,
dia date not null);

No caso da restrição abranger mais de uma coluna, temos de usar uma restrição de tabela.

Ex :

```
CREATE TABLE aluno (
    mediaCurso NUMBER(4,2) DEFAULT 0,
    media_metasExame NUMBER(4,2),
    Constraint ck_media_examens CHECK (media_metasExame < mediaCurso)
```

Restrições de tabela

→ Para garantir que uma coluna não tem valores nulos : not null

Restrições de integridade - chave Primária

- A tabela só pode ter uma chave primária!
 - A chave primária não pode conter valores nulos nem pode ter valores repetidos;

Create table Aluno (

→ A classe primária pode ser composta → usamos uma reunião de tabelas

Create table nota (

bi imkegen;

codDisc im Kegel?

mota number (4, 2),

Constraint 1K - nota Primary key (bi, coddisc)

Restrições de integridade - Chave Candidata

→ chaves candidatas podem ser definidas usando o Unique

→ não obrigam os valores a ser Not Null

```
Create table Aluno (  
    bi integer unique,
```

Restrições de integridade - chave estrangeira

↳ Uma chave estrangeira deve sempre referenciar uma chave primária ou única

```
Create table Aluno (  
    numero int primary key,  
    bi integer unique,  
    nome varchar(56) not null,  
    cursoid int references curso(curso_id)
```

atributo que é a chave primária na tabela curso

Instituição	NotaExames
ano_letivo (Pk)	ano_letivo (Fk)
numero (Fk)	nota (Pk)
cod_disc (Fk)	cod_disc (Fk)

Aluno
numero (PK)
bi
nome
cursoid (FK)

Curso
curso_id (PK)

↳ No caso da chave estrangeira ser composta (mais de uma coluna) usa-se uma restrição de tabela:

```
Create table NotasExame (  
    ano_letivo number(4),  
    numero int,  
    cod_disc number(4),  
    constraint fk_notas_exame Foreign key (ano_letivo, numero, cod_disc)  
        References instituicao (ano_letivo, numero, cod_disc))
```

;

Modifican tablas 

Alugar tabela : `DROP TABLE nome_tabela`

Acrecentar coluna : `Alter table Aluno
add (email varchar(100))`

Eliminar coluna : `Alter table Aluno
drop column email`

Alterar tipo de dados numa coluna : `Alter table Aluno
modify (email varchar(75))`

Eliminar uma constraint : `Alter table Aluno
drop constraint FK_mota_id_aluno_Aluno`

Acrecentar uma constraint : ou add constraint

Data manipulation language

Insert : inserir dados numa tabela ;

Update : atualizar // // //

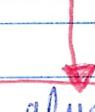
Delete : eliminar // // //

Select : recuperar dados da base de dados

Ex : `Insert into aluno Values (1,2,3,4,7777,'jáé',12.50,to_date('02.10.83','dd.mm.yy'))`

`Insert into aluno (bi, mif, nome) Values (12, 3333, 'jáé');`

`Insert into aluno Values (12,3456,'jáé',Null,Default);`

Update  Exemplos

`Update aluno`

`SET mif = 234571`

`Where nome = 'jáé'`

`Update Aluno`

`SET mediaCurso = mediaCurso * 1.2`

`Update Aluno`

`SET mediaCurso = mediaCurso * 1.2, mif = 123456`

`Where nome = 'jáé'`

Linguagem-SQL

Distinct → para forçar a eliminação de duplicados

* O SELECT também permite renomear os nomes das tabelas e atributos

WHERE

↳ permite selecionar um subconjunto de registros que satisfazem alguma condição

- and, or, not ;
- pode conter (<, >, <=, >=, ...)
- permite o uso de is null ou is not null ;

Ex: Select * from aluno

where mediacalculo = 13.2 or nome = 'IsabelAntomia' ;

→ Permite usar o operador LIKE em operações com strings;



usado para pesquisas não exatas

— (substitui 1 caractere)

% (substitui qualquer sequência de caracteres)

↳ Ex: select * from aluno where nome like 'B%u%'

Order by

→ para ordenar a resposta

Ex: order by nome ;

Ex: order by nome DESC ;

Ex: order by nome, mediacalculo DESC

Funções de agregação

→ Count : o n.º de valores

→ Sum : A soma de todos os valores

→ AVG : A média de todos os valores

→ MAX(A) : o valor máximo existente na coluna A

→ MIN(A) : o valor mínimo existente na coluna A

Exemplo: Listar o nome do aluno com média aceno mais alta

```
Select nome  
from aluno  
where mediaaceno = max(mediaaceno);
```

X)
erro correto

```
Select nome  
from aluno  
where mediaaceno = (select max(mediaaceno) from aluno);
```

Agrupamentos:

SELECT lista_atributos_a_retornar
FROM lista_tabelas_envolvidas_na_query
WHERE condições_a_verificar_em_cada_registro
GROUP BY lista_atributos_que_definem_agrupamento
HAVING condições_a_verificar_em_cada_grupo

* a cláusula having vai ter operadores de agregação

Tipos de joins

Inner joins



Outer joins



Left: alim dos registos que fazem match, os registos da tabela esquda que não existam na direita também aparecem. Quando não existe numa das tabelas os valores aparecem a null;

Right outer join: alim dos registos que fazem match, aparecem todos os restantes registos da tabela direita que não existam na esquda;

Full outer join: afazem todos os registros independentemente de fazerem ou não match.

→ Com o natural join afazem afazem os que fazem match. Tem o mesmo efeito que igualar os campos de ambas as tabelas na cláusula where.

Operações de conjuntos

- Union
 - Intersect
 - Except (em oracle é MINUS)
- } eliminam os registros repetidos

Ex: listar todos os alunos e todos os professores

SELECT numero, nome FROM aluno

Union

SELECT numero, nomeprof FROM professor;

Funções Oracle

* A função oracle NVL () permite substituir null por uma alternativa mais significativa nos resultados de uma consulta

↓ Ex:

select numero, nomeprof, NVL (caldepartamento, 'não tem')
from professor
where caldepartamento is null

* A função NVL2 permite determinar o valor retornado por uma consulta com base no fato de uma expressão especificada ser nula ou não nula.

Funções Data:

- se omisión o formato é DD-MON-YY
- Sysdate é uma função que retorna a data e hora

* TO_char (data, 'format model')

↳ converte uma data para uma string usando o formato especificado.

* TO_date (char [, 'format-model'])

- converte uma string para um formato de data;
- subtrair uma data de outra data, gerando a quantidade de dias entre elas.

* TO_extract

- permite extrair um valor a partir de uma data ou intervalo

* TO_timestamp

- converte tipos de dados (char, varchar, etc...) num valor do tipo de dados TIMESTAMP.

CASE

codigo	nome	status
123	José	B
124	Antônio	A1
125	Maria	A2
126	Dara	D
127	José	DP

→ Select que retorne o código, nome e status e significado de cada sigla:

Select cli_id,

cli_nome,

cli_status

case

when status in ('A1', 'A2') then 'Ativo'

when status = 'B' then 'Bloqueado'

• Ativo quando status A1 ou A2

• Bloqueado quando B

• Desativado quando D

• Desconhecido quando DP.

else 'legado'

END AS sigla_descricao

FROM cliente;

Case - Cláusula Where

Select first_name, last_name, country

From customers

Where

(case

when country IN ('USA', 'Canada') THEN 'North America'

when country IN ('UK', 'France') THEN 'Europe'

else 'Unknown'

END) = 'North America'

Case - Cláusula order_by

```
SELECT *
FROM locations
WHERE country_id in ('US', 'CA', 'UK')
ORDER BY country_id
CASE country_id
    WHEN 'US' THEN state
    ELSE city
END;
```

Sub Queries

Não correlacionadas → Não dependem do select exterior → 1º o select interior → Uma vez

Correlacionadas → Defende do select exterior → Execução do interior intercalada com o exterior → m vezes = m de vezes do select exterior

Para arredondar valores: ROUND (expressão matemática, 2)

$n = \text{casas decimais}$

Para remover espaços extra antes e depois da String: TRIM

↳ WHERE UPPER(TRIM(título))

LIKE '%.%.%'

Where

- Filtra linhas antes de qualquer agrupamento ou cálculo efetuado
- Usar quando queremos limitar os resultados da consulta com base em condições específicas antes de aplicar agrupamentos.

Select nome, idade

From marinheiros

Where idade > 30 ;

Group by

- Agrupa linhas com valores idênticos numa ou mais colunas para aplicar funções de agregação (ex: SUM, COUNT, AVG)
- Usar quando queremos resumir ou calcular estatísticas por grupos.

Select idade, Count (*) As total

From marinheiros

Group by idade;

↳ agrupa os marinheiros pela idade e conta quantos marinheiros existem em cada grupo

Having

- Filtra grupos criados pelo Group By, ou seja, aplica condições após o agrupamento.
- Usar quando precisamos filtrar os resultados de funções de agregação ou os próprios grupos.

Select idade, Count (*) As total,

From marinheiros

Group by idade

Having Count (*) > 2 ;

↳ mostra as idades que aparecem mais de 2 vezes na tabela, filtrando grupos com base no resultado de count.

Union

- Combina os resultados de duas ou mais consultas numa única tabela, mas elimina linhas duplicadas;

Select nome from marinheiros

UNION

Select nome from capitães;

→ Se quisermos incluir todas as linhas, incluindo as duplicadas, podemos usar o Union All.

Intersect

- Retorna apenas as linhas que estão presentes em ambas as consultas (intersecção entre as duas consultas).
- Usamos quando queremos encontrar elementos que são comuns entre as duas consultas.

Select nome from marinheiros

Intersect

Select nome from capitães

↳ Retorna os nomes que apareceram tanto na tabela marinheiros como na capitães.

Minus

- Retorna as linhas da primeira consulta que não estão presentes na segunda consulta. É a diferença entre os conjuntos de resultados.

Select nome from marinheiros

Minus

Select nome from capitães

↳ Retorna os nomes que estão na tabela marinheiros, mas não na tabela capitães.