|  |  |
| --- | --- |
| LogoEfeitrans | **Banco de Dados II**  **Lista de Fixação de Conteúdo**  Profa. Vanessa Souza |

**Questão 1:** Dado o banco Northwind, faça uma transação que aumente o preço do produto Tofu (productid = 14) em 20% a partir de **04/06/1996** e altere os pedidos desse dia em diante (tabela order\_details).

START TRANSACTION;

UPDATE northwind.products p SET unitprice= p.unitprice\*1.2 WHERE p.productId = 14;

UPDATE northwind.order\_details od

SET unitprice=(SELECT p.unitprice

FROM northwind.products p

WHERE p.productId = 14)

from northwind.orders o

WHERE od.productId = 14 and o.orderdate > '1996-06-04' and od.orderid=o.orderid;

COMMIT

**Conferindo:**

select p.productId, p.unitprice from northwind.products p WHERE p.productId = 14

select od.productId, od.unitprice

from northwind.order\_details od, northwind.orders o

WHERE od.productId = 14 and o.orderdate > '1996-06-04' and od.orderid=o.orderid;

Questão 2:

1. Crie o atributo qtdProdutos na tabela orders. O atributo deve ser inteiro e ter o valor default zero;

**ALTER TABLE northwind.orders ADD COLUMN qtdProdutos integer DEFAULT 0;**

1. Crie o atributo maisDesconto na tabela orders. O atributo deve ser inteiro e ter o valor default zero;

**ALTER TABLE northwind.orders ADD COLUMN maisDesconto integer DEFAULT 0;**

1. Atualize o atributo qtdProdutos da tabela orders com a quantidade de produtos diferentes comprados naquele pedido. Obs.: não é a soma dos itens comprados, é a quantidade de diferentes produtos adquiridos no pedido;

**UPDATE northwind.orders o SET qtdprodutos = (SELECT count(distinct od.productid)**

**from northwind.order\_details od**

**where od.orderid = o.orderid )**

**Questão 4:** Sejam as transações T1, T2 e T3, as quais desempenham as seguintes operações:

1. durabilidade e consistência. B) persistência e automação. C) isolação e atomicidade.
2. durabilidade e atomicidade. E) consistência e persistência.

**Questão 3:** Considere:

1. Se uma transação é concluída com sucesso (operação commit bem sucedida), então seus efeitos são persistentes. => durabilidade
2. Ou todas as ações da transação acontecem, ou nenhuma delas acontece. => atomicidade.

As propriedades (I) e (II) das transações em SGBDs, significam, respectivamente:

T1 : **Adicionar 1 ao A** T2 : **Duplicar** e T3 : Exibir A na tela (crie uma função para isso), e então **fixar A em 1**. Suponha que as transações T1, T2 e T3 possam ser executadas concorrentemente. Se **A tiver um valor inicial de zero** e as transações forem **executadas de forma serial** (uma após a outra), quantos resultados corretos possíveis teremos? Listá-los.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ordem da Transaçoes**  **1ª 2ª 3ª** | | | **Processamento** | **Resultado** |
| T1 | T3 | T2 | T1 = 1, T2 = 2, T3 = 1 | **1** |
| T1 | T2 | T3 | T1 = 1, T3 = 1, T2 = 2 | **2** |
| T2 | T1 | T3 | T2 = 0, T1 = 1, T3 = 1 | **1** |
| T2 | T3 | T1 | T2 = 0, T3 = 1, T1 = 2 | **2** |
| T3 | T2 | T1 | T3 = 1, T2 = 2, T1 = 3 | **3** |
| T3 | T1 | T2 | T3 = 1, T1 = 2, T2 = 4 | **4** |

* Teremos um total de 6 resultados possiveis, onde 4 deles são diferentes entre si.

**Questão 5:** Considere um disco com bloco de 512 bytes. Um ponteiro para bloco de 6 bytes e um ponteiro para registro de 7 bytes. Um arquivo contém 75000 registros de tamanho fixo de FILMES. Cada registro tem os seguintes atributos : FilmeID (4 bytes), DiretorID (4 bytes), CategoriaID (4 bytes), Título (35 bytes), AnoLancamento(4 bytes), Duracao (4 bytes), Resenha (50 bytes). Um byte adicional é usado como marcador de deleção do registro. O atributo **FilmeID** é a chave primária da tabela. O atributo **Título** é uma chave candidata.

**Dados:**

* Tamanho do bloco = 512 bytes
* Ponteiro do bloco (Índ. Primário) = 6 bytes
* Ponteiro de registro (Índ. Secundário) = 7 bytes
* Registros no arquivo = 75.000
* Tamanho registro=4+4+4+35+4+4+50=105 bytes

1. Suponha que o arquivo é sequencial. Calcule:
   1. O tamanho do índice primário.

**Info:**

PK\_FilmeID = 4 bytes;

Ponteiro do bloco (Índ. Primário) = 6 bytes

* 1 entradra do Índice Primário = Tamanho\_FilmeID + Tamanho\_Ponteiro

1 entradra do Índice Primário = 4+ 6 = 10 bytes

* Quantidade de registros por bloco: RB

RB = Tam\_Bloco / Tam\_Registro

RB = 512/105 = 4,8

* RB = 4 registros por bloco
* Quantidade de Blocos total Necessário: BT  
   1 bloco ----- 4 registros

BT ---------- 75.000 registros

* BT = **18.750 blocos** serão utilizados para guardar todos os registros
* Quantidade de indice primario por bloco: IPB

IPB = Tamanho do bloco/ Tamanho da entrada de Índice Primário

IPB = 512/10 = 51,2 registros por bloco

* IPB = 1 bloco de indice primário cabem 51 indices primários
* Quantidade de Blocos para armazenar o índice primário: BIP
* Como se trata de um índice primário esparço, quantidade indice = quantidade de blocos
* serão necessários **18.750**  **blocos** indices

1 bloco ----- 51 indices

BIP ---------- **18.750** indices

BIP = BT/ IPB

BIP = **18.750** /51 = 367,64

* **BIP = 368 blocos são utilizados para os indices primários**

1. O tamanho do índice sobre o atributo AnoLancamento.

**Info:**

AnoLancamento = 4 bytes;

Ponteiro de registro (Índ. Secundário) = 7 bytes

* Tam\_entrada\_Ind\_AnoLancamento = Tamanho\_AnoLancamento + Tamanho\_Ponteiro\_Sec

Tam\_ entrada\_Ind\_AnoLancamento = 4 + 7 = 11 bytes

Quantidade de indice secundario por bloco: ISB

ISB = Tamanho do bloco/ Tamanho da entrada Índice Secundario

ISB = 512/11 = 46,54

* NRSB = 46 indices por bloco

* Blocos para armazenar o índice secundário:BIS
* Como se trata de um índice secundário denso, serão necessários **75.000** **Indices**.

1 bloco ----- 46 indices

BIS ---------- 75.000 indices

BIS = N° total de indice / ISB

BIS = 75.000 /46 = 1.630,43

* **QBIP = 1631 blocos são utilizados para o indice secundario**

1. Suponha que o arquivo é heap. Calcule:

**Arquivo heap -> Todos os indices densos (primario e secundarios)**

* 1. O tamanho do índice primário.
* Tamanho de 1 entrada do Índice Primário = 4+ 7 = 11 bytes
* Quantidade de indice primario por bloco: IPB

IPB = Tamanho do bloco/ Tamanho da entrada Índice Primário

IPB = 512/11 = 46,54 registros por bloco

* IPB = 1 bloco de indice primário cabem 46 indices primários
* Como se trata de um índice primario denso o numero de indice = numero de registros
* serão necessários **75.000** **Indices**.
* Numero de blocos de incides primario necessarios:

BIP = N° total de indices/ IPB

BIP = 75.000 /46 = 1.630,43

* **BIP = 1.631 blocos são utilizados para os indices primários**
* Tamanho necessario para o indice primario em bytes:

1 bloco ----------------- 512 bytes

**1.631** blocos ------------ Qb

Qb = 835.072 bytes/1024bytes

* São necessários 3835.072 bytes ou 815,5 kb para o índice secundario
  1. O tamanho do índice sobre o atributo Título. -> denso
* Tamanho de 1 entrada do Índice Primário:

Tam\_entrada\_Ind\_Titulo = Tamanho\_titulo + Tamanho\_Ponteiro\_Sec

Tam\_ entrada\_Ind\_Titulo = 35 + 7 = 42 bytes

* Quantidade de indice secundario por bloco: ISB

ISB = Tamanho do bloco/ Tamanho entrada Índice Primário

ISB = 512/42 = 12,19

* NRIPB = 1 bloco de indice secundario cabem 12 indices
* Quantidade de blocos para indice secundário: quuantidade de indice igual quantidade de registros

1 bloco ----- 12 indices

BIS ---------- 75.000 indices

BIS = N° total de indices / ISB

BIS = 75.000 /12 = 6.250

* **BIS = 6.250 blocos são utilizados para os indices secundario**
* Tamanho necessario para o indice primario em bytes:

1 bloco ----------------- 512 bytes

6.250 blocos ------------ Qb

Qb =**3.200.00 bytes**/1024 = 3.125 Kbytes

* São necessários **3.125 Kbytes** bytes para o índice secundario
  1. Qual alternativa pode ser utilizada para reduzir o tamanho do índice sobre o atributo Título?
  + Utilizar apenas as 3 ou 4 letras iniciais do titulo para fazer o índice.

1. Por que podemos ter no máximo um índice primário ou *clustering*, mas diversos índices secundários para um arquivo de dados sequencial?

Um arquivo pode possuir no máximo um índice primário ou clustering, pois são baseados no campo de ordenação física de um arquivo, ou seja, sua chave primária que é única. E os índices secundários podem ser especificados sobre qualquer campo não ordenado de um arquivo. **A diferença entre índice secundário e índice primário está no fato de que o índice secundário não determina a colocação de registros no banco de dados**. O que permite que um arquivo possa ter diversos índices secundários além de um único índice primário.

1. Ilustre um cenário que use o índice primário criado e outro que não use. Gere as consultas SQL nos dois casos. OBS. Em ambos os casos, a consulta deve utilizar a PK.

**Que utiliza o indice:** Querendo saber o nome do filme sabendo que o id = 10 e categoriaID=4 :

Select Título from filmes where filmeID = 10 and categoriaID=4;

**Que não utiliza:** Querendo saber quais são os filmes que foram lançados 1998 ou quais são da categoria 4 ou quais tem categoriaID>100

Select filmeID, Título from filmes where AnoLancamento= 1998 or CategoriaID=4 or filmeID = 100;

**Observa-se que para porta logica and o indice é utilizado mas para or não.**

**Questão 6:** Um arquivo sequencial armazena 20.000 registros de ALUNO de tamanho fixo. Cada registro tem os seguintes campos: Nome (30 bytes), Cpf (9 bytes), Endereço (40 bytes), Telefone (10 bytes), Data\_nascimento (8 bytes), Sexo (1 byte), Dep\_princ (4 bytes), Dep\_sec (4 bytes), Tipo\_aluno (4 bytes) e Titulo\_academico (3 bytes). Um byte adicional é usado como um marcador de exclusão. A chave primária da tabela aluno é CPF. O arquivo é armazenado num disco, cujo bloco tem 512 bytes.

* São **20.000 registros** de dados, onde um registro ocupa:

Tam\_Registro = Tam\_Nome + Tam\_CPF + Tam\_Endereço +Tam\_Telefone +

Tam\_Data\_Nascimento + Tam\_Sexo + Tam\_Dep\_princ + Tam\_Dep\_sec +

Tam\_Tipo\_aluno + Tam\_Titulo\_academico + Tam\_Marcador\_Exclusao

**Tam\_Registro** = 30 + 9 + 40 + 10 + 8 + 1 + 4 + 4 + 4 + 3 + 1 = **114 bytes**

* Cada **bloco** do disco armazena **512 bytes**

1. Se um ponteiro ocupa 4 bytes, quantos bytes são necessários para o índice primário?

* Quantidade de Registro por blobo: RPB

RPB = tam\_bloco / tam\_registro

RPB =512/114 = 4,49 = 4 registros por bloco

* Quantidade de blocos dos registros BT

1. bloco ------------ 4 registros

BT -------------------- 20000 registros

* BT = 5000 blocos de dados
* Como se trata de um índice primário, quantidade de indice = quatidade de blocos de registros:
* Serão necessários 5000 índices.
* Tamanho de 1 entrada do Índice Primário:

A entrada de índice ocupa = 9 de chave + 4 de ponteiro = 13 bytes

* Quantidade de indice primario por bloco: IPB

IPB = 512(tam\_bloco) / 13 bytes (tam\_entrada\_indice\_prim)

IPB = 39,38 = 39 registros por bloco.

* Numero de bloco do indice primario BIP:

1 bloco ----------------- 39 registros

QBIP --------------------- 5.000 registros

QBIP = 128,2 = 129 blocos

* São necessários 129 blocos para índice primário
* Tmanho do bloco do indice primario em bytes Qb:

1 bloco ----------------- 512 bytes

1. blocos ------------ Qb

Qb = 66.048 bytes

* São necessários 66.048 bytes para o índice primário.

1. Qual o tamanho de um índice secundário criado sobre o atributo Sexo (1), cujo ponteiro ocupa 8 bytes?

* O registro de índice ocupa 9 bytes (1 do atributo secundario e 8 de ponteiro)
* Quantidade de indice secundario por bloco: ISB

ISB= 512/9 = 56,88 = 56 índices secundários cabem em 1 bloco.

* Como se trata de um índice secundario (denso) serão necessários 20.000 indices, um para cada registro na tabela.
* Numero de blocos por indice secundario: BIS

1 bloco ----------------- 56 registros

BIS ------------------------- 20000

BIS = 357,14 = 358 blocos

* São necessários 358 blocos para o índice secundario
* Tamanho em bytes dos indices

1 bloco ----------------- 512 bytes

358 blocos ------------ Qb

Qb = 183.296 bytes

* São necessários 183.296 bytes para o índice secundario = 179KB

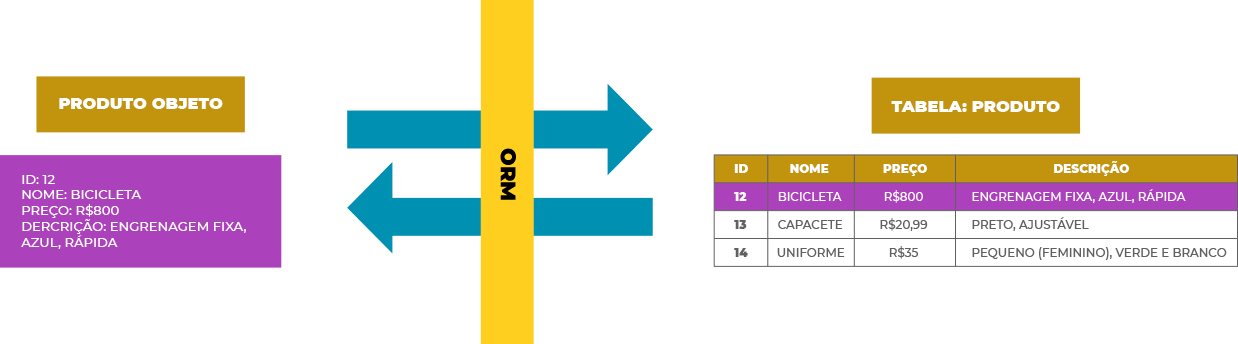
1. Sobre esses índices acima mencionados, escreva em SQL e justifique uma consulta que usaria o índice e outra que não usaria. **ATENÇÃO** : A consulta precisa utilizar o atributo indexado.

**Usando o indice**: Buscar o titulo academico dado o cpf e nome como condição

Select titulo from aluno where cpf=’xxxx’ and nome= ‘pedro’ retona o titulo do aluno pedro

**Sem usar o indice:** buscar o telefone dado como condição cpf ou data de nacimento

Select telefone from aluno where cpf=’xxxx’ or datanacimento= ‘1998-11-23’ retorna o telefone do aluno com o cpf “xxxx” e o telefone do aluno com a data de nacimento 23/11/98

**Questão 7:** Considerando os conceitos aprendidos na disciplina de banco de dados, discorra sobre a figura abaixo.

O mapeamento Objeto-Relacional (ORM) foi desenvolvida para reduzir a impedância entre os modelos OO e MR. A figura apresenta os 3 componentes do ORM: 1- o **Modelo Orientado a objetos** em que **os dados são representados na aplicação** representada pelo ‘Produto objeto”. 2- A **Persistência Lógica** que **traduzirá o modelo OO de maneira que eles sejão armazenados na persistência física, e vice-versa**. E 3- a **Persistência Física** que **representa o Modelo relacional** “Tabela:produto” **em que os dados são armazenados no banco**. Assim sendo o ORM tem como função traduzir o “produto objeto bicicleta” da aplicação para o registro da “tabela produto” do banco relacional e vice versa.

**Questão 8:** Considere a relação EMPREGADO (NumeroEmp, RG, nome, sobrenome, salario, endereço, departamento), em que o atributo grifado corresponde à chave primária da relação. Considerando o SGBD PostgreSQL e a relação EMPREGADO descrita:

1. Escreva em SQL os comandos que devem ser utilizados para implementar corretamente a segurança na tabela EMPREGADOS, de forma que o usuário A1 tenha apenas direito de realizar consultas nesta tabela; o usuário A2, além de consultas, possa também realizar inclusões, alterações e exclusões de dados, além de propagar os privilégios para outros usuários; o usuário A3 tem acesso de consulta ao total de empregados por departamento. Considere que estes usuários ainda não possuam nenhum privilégio sobre a tabela EMPREGADOS.

CRIAÇÃO DE USUÁRIOS PARA REALIZAR LOGIN/AUTENTICAÇÃO:

CREATE USER A1 WITH PASSWORD ‘SENHA’

CREATE USER A2 WITH PASSWORD ‘SENHA’

CREATE USER A3 WITH PASSWORD ‘SENHA’

PARA ATRIBUIR OS DIREITOS:

Dar direito sobre o schema aos 3 empregados:

GRANT USAGE ON SCHEMA NOME\_DO\_SCHEMA TO A1, A2, A3

Dar direito ao A3 de acesso de consulta ao total de empregados por departamento:

CREATE VIEW dadosA3 AS

(SELECT departamento, count (numeroEmp) as total\_empregado

FROM empregado

GROUP BY departamento);

/\*Conceda o privilégio de leitura sobre a View para a role ‘A3’\*/

GRANT SELECT ON dadosA3 TO A3;

Dar direiro ao usuário A1 para realizar consultas na tabela Empregado:

GRANT SELECT ON EMPREGADO TO A1;

Dar direiro ao A2 de consultar, incluir, alterar e excluir e, ainda, de propagar os privilégios para outros usuários:

GRANT SELECT,INSERT,UPDATE,DELETE ON EMPREGADO TO A2 **WITH GRANT OPTION**;

1. Qual o nível de segurança implementado na letra (a)? Tal nível é suficiente para garantir a segurança do banco?

Nível de Sistema de Banco de Dados. Não, pois no proprio nível o usario A2 conseguirá dar privelegios a qualquer outro usuário e este por sua vez poderá tambem atribuir privelegios a outros e assim sucessivamente. Deve se atribuir poderes(direitos) aos usuários com parcimonia (sem with grant option). E para garantir a segurança, além de limitar o acesso ao banco para não dar super poderes ao usuário deverá tambem garantir a segurança do banco pelos demais níveis : **Físico (como restringir a acessibilidade dos equipamentos tecnologicos); Sistema Operacional (como garantir a atualização do sistema, antivirus, firewall, programas e a sua restrição no que tange ao acesso remoto) e Humano (como por exemplo o usuário não deveria passar sua senha e nem dar acesso a outro usuario seja qual for o motivo)**

1. Qual a diferença entre segurança e integridade?

**A Segurança** refere-se a medidas **contra acessos maldosos**, sendo um conceito **relacionado a danos causados de forma intencional no banco** de dados. Já a **Integridade** refere-se ao ato de **evitar a perda acidental de consistência.**

--------------------------------------------------------------------------------------------------------------------------

**Resumo de bd2**

**Consistência:** utiliza-se de regras de integridade para que os dados sejam assegurados. **Segurança :** Refere-se as medidas contra acessos maldosos. **Integridade**: Refere-se ao ato de evitar a perda acidental de consistência. Segurança é um conceito relacionado a danos causados de forma intencional no banco de dados. Princípios da Segurança da Informação: **Integridade, Disponibilidade e Confidencialidade**. **Violações de Integridade:** Quebras durante o processamento de transações. Anomalias causadas por acesso concorrente ao banco de dados. Anomalias causadas pela distribuição de dados sobre diversos computadores. Um erro lógico que viola a suposição de que as transações preservam as restrições de consistência do banco de dados. **Violações de Segurança**: Leitura não-autorizada de dados (roubo de informação). Modificação não-autorizada de dados. Destruição não-autorizada de dados. A fim de proteger o banco de dados, medidas de segurança precisam ser tomadas em diversos níveis: ♦ Físico ♦ Sistema Operacional ♦ Sistema de Banco de Dados ♦ Humano. **Nível Físico**: O local ou locais onde os sistemas de computador estão localizados precisam estar fisicamente protegidos contra assaltos ou intrusos. **Nível de Sistema Operacional**: A fraqueza na segurança do sistema operacional pode servir como um meio para acesso não autorizado ao banco de dados. Uma vez que quase todos os sistemas de banco de dados permitem o acesso remoto através de terminais ou redes, a segurança no nível do software dentro do sistema operacional é tão importante quanto no nível físico. **Nível de Sistema de Banco de Dados.** Alguns usuários de banco de dados podem estar autorizados a fazer o acesso apenas a uma porção limitada do banco de dados. A outros usuários pode ser permitida a formulação de consultas, mas proibida a modificação de dados. É responsabilidade do sistema de banco de dados assegurar que essas restrições não sejam violadas. **Nível Humano**: Os usuários devem ser cautelosamente autorizados para reduzir a chance de qualquer usuário dar acesso a um intruso em troca de suborno ou outros favores. A segurança de todos os níveis precisa ser mantida a fim de garantir a segurança do banco de dados. A **autenticação** é o processo pelo qual o servidor de banco de dados estabelece a identidade do cliente e, por extensão, determina se o aplicativo cliente (ou o usuário executando o aplicativo cliente) tem permissão para se conectar com o nome de usuário que foi informado.

**Transação** **é uma unidade lógica de trabalho, envolvendo diversas operações de bancos dados**. Do ponto de vista do SGBD, uma transação **é uma sequência de operações que são tratadas como um bloco único e indivisível** (atômico) no que se refere à sua recuperação. Assegura que atualizações inacabadas ou atividades corrompidas não sejam executadas no banco de dados. Está envolvida com os comandos Update, Delete e Insert. Leituras não comprometem a consistência dos dados no banco. Atualizações sim!!! A integridade de uma transação depende de 4 propriedades, conhecidas como ACID: **Atomicidade, Consistência, Isolamento e Durabilidade**. **Atomicidade**: **Princípio do “Tudo ou Nada”.** Ou todas as operações da transação são efetivadas com sucesso no BD, ou nenhuma delas será efetivada. Preserva a integridade do banco. É Responsabilidade do subsistema de recuperação contra falhas (recovery) do SGBD. **Isolamento :** Uma transação não deve sofrer **interferência** de outras que estejam sendo executadas de forma **concorrente**. Responsabilidade conjunta subsistema de controle de concorrência (scheduler) do SGBD. **Durabilidade:** Deve garantir que as modificações realizadas por uma transação que concluiu com sucesso **persistam** no BD. Nenhuma falha posterior ocorrida no BD deve perder essas modificações. Responsabilidade do subsistema de recuperação contra falhas (recovery) do SGBD. **Consistência**: **Uma transação sempre conduz o BD de um estado consistente para outro estado também consistente.** Responsabilidade conjunta : DBA, subsistema de recuperação contra falhas e subsistema de controle de concorrência. As transações transferem blocos de informações do disco para a memória principal e depois os devolvem para o disco Os blocos residentes no disco são chamados blocos físicos, e os residentes na memória, de blocos de buffer. Os movimentos de blocos entre o disco e a memória são feitos por duas operações: **Input(X):** **Transfere do disco para a memória o bloco físico X** e **Output(X):** **Transfere da memória para o disco o bloco de buffer X** e substitui o bloco físico apropriado. As transações interagem com o sistema de banco de dados transferindo dados de variáveis de programa para o banco de dados (arquivo!) e, do banco de dados para as variáveis de programa. Esta transferência de dados é realizada usando outras duas operações: **Read(X, xi):** Atribui o valor de X para a variável local xi. Se o bloco no qual X reside não estiver na memória, então emite input(X). Atribui o valor do bloco de buffer X para xi. **Write(X,xi)**: Atribui o valor da variável local xi para o bloco de buffer X. Se o bloco no qual X reside não estiver na memória, então emite input(X). Atribui xi para o valor de X do bloco de buffer. Ex.: 1. read(A,a1) 2. a1:= a1 – 100 3. write (A,a1) 4. read(B,b1) 5. b1:=b1+ 100 6. write(B,b1).

**ORM:** **Mapeamento objeto/relacional é a persistência automatizada e transparente dos objetos em uma aplicação Orientada a Objetos para as tabelas de um banco de dados relacional**, utilizando metadados que descrevem o mapeamento entre os objetos e o banco de dados. Mapeamento Objeto-Relacional foi desenvolvida para reduzir a impedância entre os modelos OO e MR. Uma solução ORM consiste nas seguintes 4 partes: 1-Uma API para realizar operações CRUD básicas em objetos de classes persistentes. 2-Uma linguagem ou API para especificar consultas que se referem às classes ou às propriedades das classes. 3-Uma facilidade para especificar o metadado de mapeamento. 4-Uma técnica para que a implementação ORM interaja com objetos transacionais para executar funções de otimização. O mapeamento OR tem 3 componentes: **Modelo Orientado a objetos:** Modelo OO em que os dados estão representados na aplicação. **Persistência Lógica:** traduz o modelo OO para a maneira que eles serão armazenados na persistência física, e vice-versa. **Persistência Física**: Modelo relacional em que os dados serão armazenados.Em uma aplicação que use ORM, devemos: 1. Gerar o modelo de classes (model): Mapeamento entre as tabelas do banco de dados e as classes da aplicação e 2. Manipular os dados de forma Orientada a Objetos. **Frameworks ORM:** Definem o modo como os dados serão mapeados entre os ambientes, como serão acessados e gravados. Isso diminui o tempo de desenvolvimento, uma vez que não é necessário desenvolver toda essa parte.

**DAO:** Data Access Object Pattern ou DAO pattern. Esse padrão introduz uma camada de abstração entre a camada de lógica de negócios e a camada de armazenamento persistente. Usado para abstrair e encapsular todo acesso ao banco-Camada de persistência. É usado para separar APIs ou operações de acesso a dados de baixo nível de serviços de negócios de alto nível. DAO não é transacional! Transações fazem parte da aplicação e devem estar declaradas no model.

**Índices** são, portanto, **estruturas de dados** auxiliares cujo único propósito é **tornar mais rápido o acesso a registros baseados em certos campos**, chamados campos de indexação. Cada índice é formado pelo valor de um atributo chave do registro e pela sua localização física no interior do arquivo. A estrutura com a tabela de índices é também armazenada e mantida em disco. Cada estrutura de índice está associada a uma chave de busca particular. **Ordenados:** Baseiam-se na ordenação dos valores. **Hash:** Baseiam-se na distribuição uniforme dos valores determinados por uma função (função de hash). Classificação de Índices Ordenados: **Considerando a quantidade de entradas: Denso ou Esparso**. **Considerando a organização do arquivo: Primário ou Clustering ou Secundário.** **Considerando os níveis de indirecionamento: Mononível ou Multinível**. **Índice denso:** **Uma entrada de índice para cada registro da tabela**. **Índice esparso:** **Uma entrada de índice para cada bloco do arquivo**. Um índice esparso utiliza menos espaço para armazenamento ao custo de um tempo um pouco maior para encontrar um registro dada sua chave. No índice esparso, inserções e remoções são menos custosas quando comparadas ao índice denso. Em **arquivos SEQUENCIAIS, o índice primário é sempre ESPARSO**. **Arquivos heap não suportam indices esparsos, assim o indice é sempre denso. Indice secundarios são sempre densos tanto em arquivos heap quanto em sequenciais**. **Índices Secundários:** Cada alteração na tabela deve ser refletida no banco. O arquivo de índice tem a mesma quantidade de registros da tabela e ocupam espaço em disco. Se a tabela heap não ordena os dados fisicamente pela chave primária, como se dá o índice de chave primária? Arquivos Heap são listas de registros não ordenados de tamanho variado. Uma vez que não há ordenação, para localizar uma informação em um Heap é necessário realizar sempre um “table scan”. Heap é uma tabela sem índice clusterizado (esparso). O índice é do tipo UNIQUE. Todo índice UNIQUE é secundário. Os índices PRIMÁRIOS são criados automaticamente pelo SGBD, a partir do comando CREATE TABLE. Todos os índices que criamos no banco são SECUNDÁRIOS e, portanto, DENSOS. Utilizando arquivos sequenciais: Existe índice primário (esparso) para a chave primária da tabela e Existe índice secundário (denso) para os demais campos . Utilizando arquivos heaps: Existe apenas índice secundário, mesmo para a chave primária. Utilizar índices melhoram a eficiência das buscas pois estes diminuem o número de acessos ao disco. Num índice primário, precisa necessariamente que seja um atributo EXCLUSIVO, e o único exclusivo é a chave primária (PRIMARY KEY). A chave candidata não é ordenada, e pode ser usada na criação de índices secundários