Indexação

Conceitos Básicos
Índices
Índices Ordenados
Índice Clusterizado (Primário)
Índice Não Clusterizado (Secundário)
Estrutura de Índices Ordenados
Árvore Binária
Criação de Índices no Oracle
Uso de Constraints e Índices
Checklist para Criação de Índices
Índices Baseados em Funções

1. Conceitos Básicos

À medida que os bancos de dados tornam-se cada vez maiores por causa do contínuo crescimento no volume de dados armazenados, a importância e **necessidade de acessos eficientes** aos dados tornam-se um desafio para o Administrador de Banco de Dados (DBA).

Modernos Sistemas de Gerência de Bancos de Dados (SGBDs) oferecem uma **vasta gama de recursos e funcionalidades** que podem ser exploradas pelo DBA para melhorar o desempenho global do banco de dados.

Um destes recursos que o DBA pode explorar é a possibilidade de **criação de índices** sobre as relações. Os índices podem melhorar o desempenho das consultas uma vez que oferecem um caminho alternativo de acesso aos dados.

1. Conceitos Básicos

Índices são **estruturas de acesso auxiliares associadas às relações** (tabelas) e que têm por **objetivo aumentar o desempenho na execução de consultas**. Tipicamente, um índice permite que um registro seja encontrado sem que seja necessário acessar toda a relação (arquivo de dados).

Quando uma relação não possui índices definidos, o SGBD deve percorrer toda a relação para localizar os dados desejados (também chamado de **"*full table scan*"**).

Isso pode tornar o tempo de resposta de consultas sobre relações com milhares de tuplas bastante alto, sendo que a **finalidade de um índice** é permitir a **rápida determinação do endereço de um registro do arquivo**, dado um argumento de pesquisa.

O endereço identifica a posição onde está armazenada a tupla (linha), na memória secundária.

2. Índices

Um arquivo de índice consiste em registros (chamados entradas de índice) na forma:

Cada estrutura de índice está associada a uma chave de busca.

Chave de busca: atributo ou conjunto de atributos para pesquisar registros em um arquivo.
Ponteiro: endereço onde está armazenada a tupla (linha), na memória secundária.

Arquivos de índice normalmente são muito menores do que o arquivo original.

Arquivo de Dados

	CHAVE	REGISTRO
0	João	...
1	Maria	...
2	Antonio	...
3	Rosa	...
4	Carlos	...
5	Beto	...
6	Francisco	...

Estrutura de Índice

CHAVE	ENDEREÇO
Antonio	2
Beto	5
Carlos	4
Francisco	6
João	0
Maria	1
Rosa	3

2. Índices

Métricas de Avaliação para Índices:

1) Tipos de acesso admitidos com eficiência:
    a) Registros com um valor especificado no atributo (campo = <valor>);
    b) Registros com um valor de atributo compreendido em um intervalo especificado de valores (atributo campo <valor1> and <valor2>).
2) Tempo de alteração;
3) Tempo de inserção;
4) Tempo de exclusão;
5) Sobrecarga de espaço adicional para implementação.

3. Índices Ordenados

Em um índice ordenado, as entradas de índice são armazenadas e classificadas pelo valor da chave de busca.

1. Índice Clusterizado (Primário):

Em um arquivo (tabela) ordenado seqüencialmente, é o índice cuja chave de busca especifica a ordem seqüencial do arquivo.

A chave de busca de um índice primário é normalmente (mas não necessariamente) a chave primária.

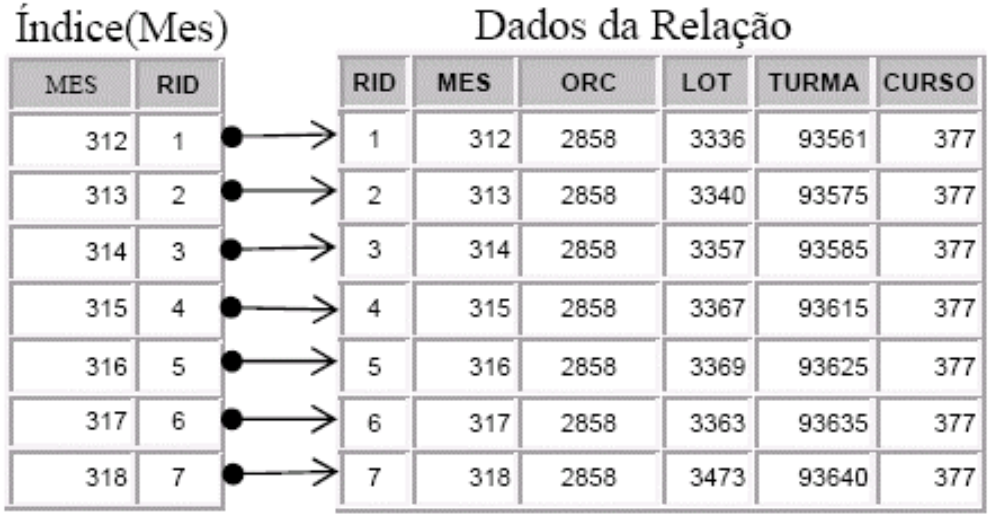
2. Índice Não Clusterizado (Secundário):

Um índice cuja chave de busca especifica uma ordem diferente da ordem seqüencial do arquivo.

4. Índice Clusterizado (Primário)

O índice cluster **é um arquivo ordenado** e composto por dois campos. O primeiro campo corresponde a **chave usada para a indexação** e o segundo campo é um **ponteiro para a tupla** correspondente no arquivo de dados.

Abaixo a representação de um índice cluster onde o arquivo de dados está ordenado de acordo com a chave do índice (atributo *mês*). A coluna RID representa o ponteiro que armazena a localização física do registro no arquivo de dados.



MES	RID
312	1
313	2
314	3
315	4
316	5
317	6
318	7

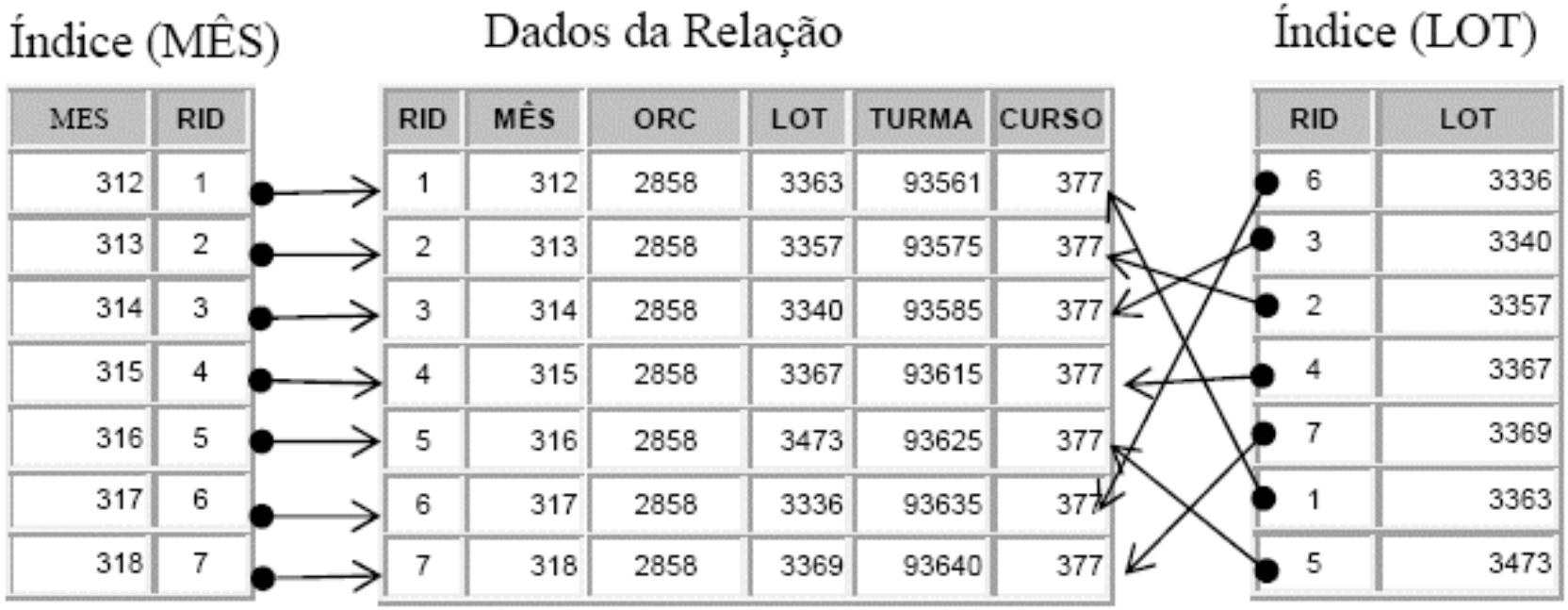
RID	MES	ORC	LOT	TURMA	CURSO
1	312	2858	3336	93561	377
2	313	2858	3340	93575	377
3	314	2858	3357	93585	377
4	315	2858	3367	93615	377
5	316	2858	3369	93625	377
6	317	2858	3363	93635	377
7	318	2858	3473	93640	377

5. Índice Não Clusterizado (Secundário)

Uma limitação das relações tradicionais reside no fato de que elas só podem estar fisicamente ordenadas considerando apenas um índice, ou seja, uma dimensão. Quando um novo índice é definido sobre a relação, este não terá a propriedade de ser índice cluster.

Neste tipo de índice, observa-se que a ordem dos dados não corresponde ao ordenamento da chave do novo índice.

Abaixo é exibida a relação e a adição de um novo índice (no atributo *LOT*).



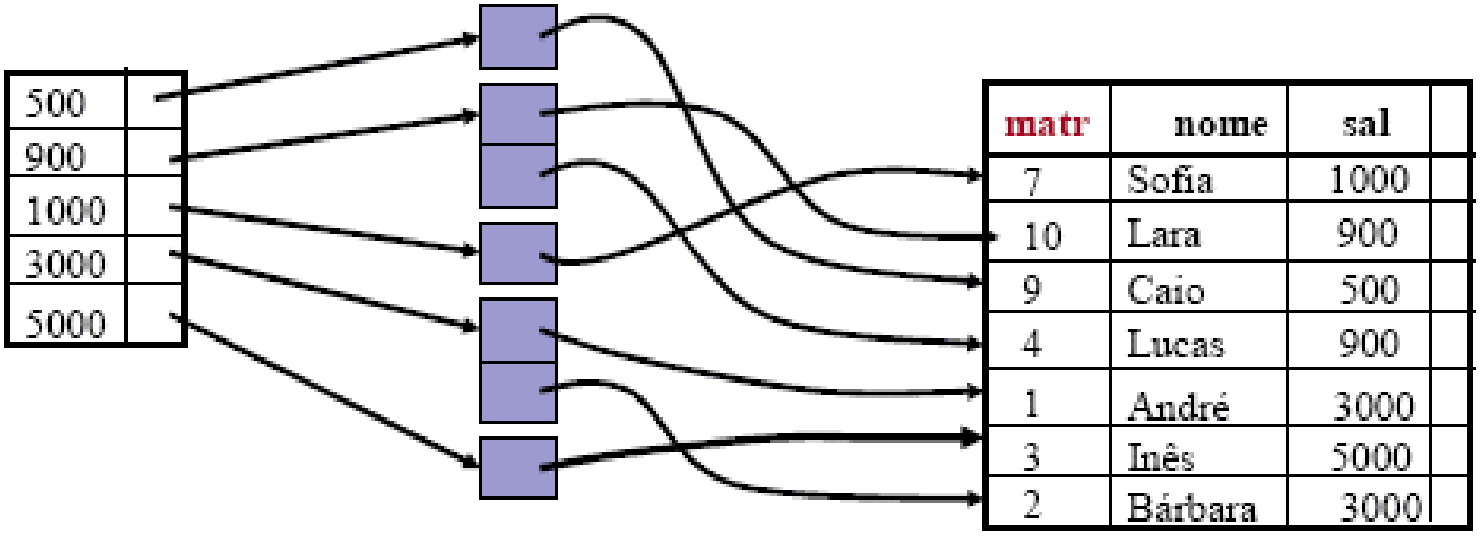
MES	RID
312	1
313	2
314	3
315	4
316	5
317	6
318	7

RID	MÊS	ORC	LOT	TURMA	CURSO
1	312	2858	3363	93561	377
2	313	2858	3357	93575	377
3	314	2858	3340	93585	377
4	315	2858	3367	93615	377
5	316	2858	3473	93625	377
6	317	2858	3336	93635	377
7	318	2858	3369	93640	377

RID	LOT
6	3336
3	3340
2	3357
4	3367
7	3369
1	3363
5	3473

5. Índice Não Clusterizado (Secundário)

- Índice Secundário (cont.)
  - Chave de busca definida sobre atributo não chave
    - Nível extra de indireção



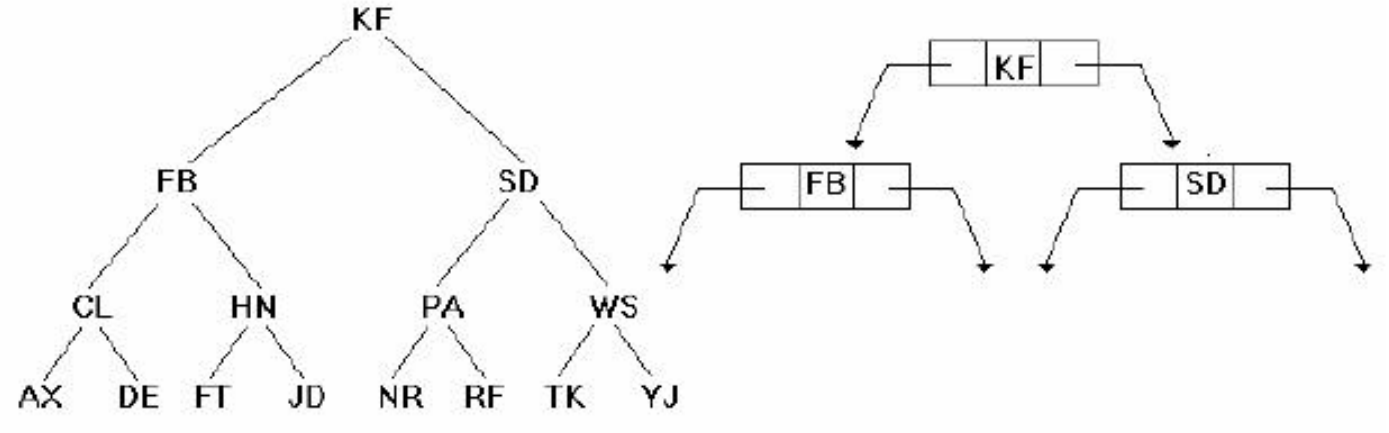
6. Estrutura de Índices Ordenados

Bayer and McGreight, em 1972, publicaram o artigo: *"Organization and Maintenance of Large Ordered Indexes"*.

Em 1979, o uso de **árvores-B** já era praticamente o padrão adotado em sistemas de arquivos de propósito geral para a manutenção de índices em bases de dados.

Os registros são mantidos num arquivo, e **ponteiros (esq e dir) indicam onde estão os registros filhos**. Esta estrutura será mantida em memória secundária: os ponteiros para os filhos dariam o número dos registros correspondentes aos filhos e sua localização no arquivo.

7. Estrutura de Índices Ordenados (Árvore Binária)



Raiz = 9

	key	filho esq.	filho dir.
0	FB	10	8
1	JD		
2	RF		
3	SD	6	15
4	AX		
5	YJ		
6	PA	11	2
7	FT		

	key	filho esq.	filho dir.
8	HN	7	1
9	KF	0	3
10	CL	4	12
11	NR		
12	DE		
13	WS	14	5
14	TK		

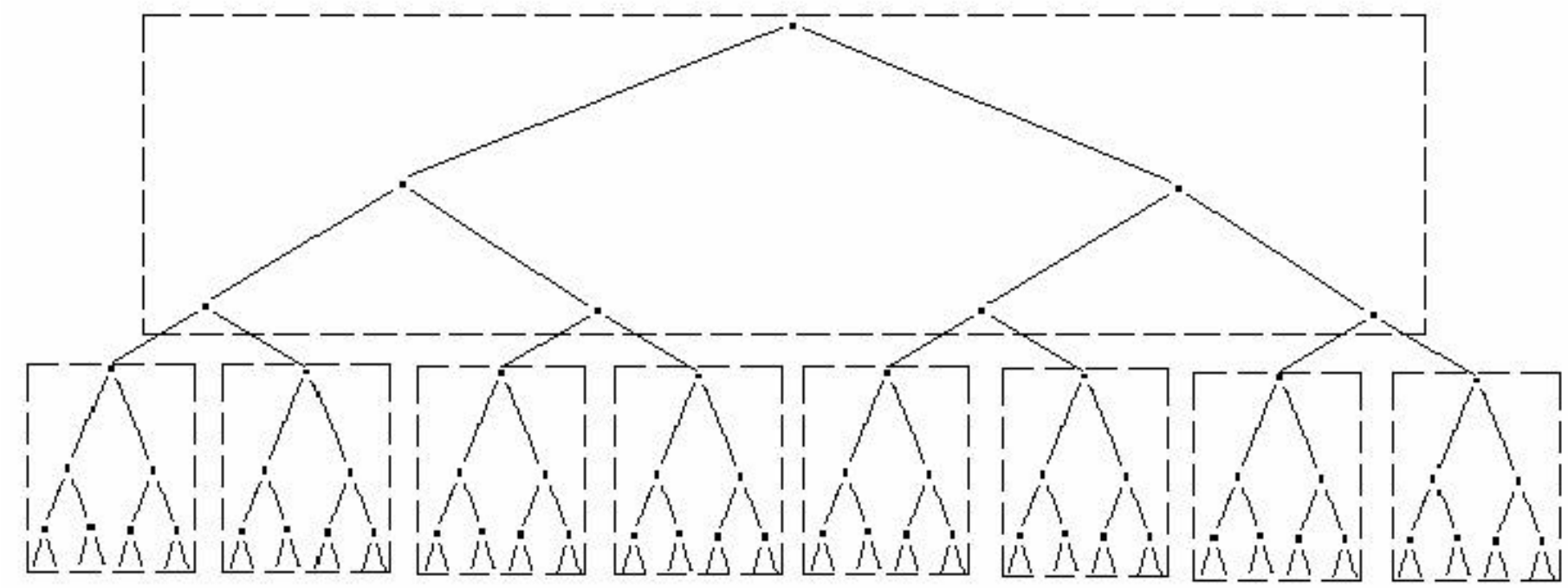
7. Estrutura de Índices Ordenados (Árvore Binária)

Para uma **árvore binária**, o **número máximo de comparações para localizar uma chave** em uma árvore com **N chaves** é igual à altura da árvore, dada por **log2 (N+1)**.

Portanto, para cerca de 1.000.000 de chaves, uma busca irá percorrer até 20 níveis da árvore (pesquisa).

Cada página pode conter um número bastante grande de registros, e se por acaso o próximo registro a ser recuperado estiver na mesma página, você economizou um acesso a disco.

7. Estrutura de Índices Ordenados (Árvore Binária)



A divisão de uma árvore binária em páginas é ilustrada na figura acima. Nessa árvore de 9 páginas, quaisquer dos 63 registros podem ser acessados em, no máximo, 2 acessos.

1. **Árvore Binária: log2 (N+1) = log2 (63+1) = 6 pesquisas**
2. **Árvore Binária Paginada = 2 pesquisas**

8. Criação de Índices no Oracle

Sintaxe para criação:

CREATE INDEX índice ON tabela (coluna[, coluna]...);

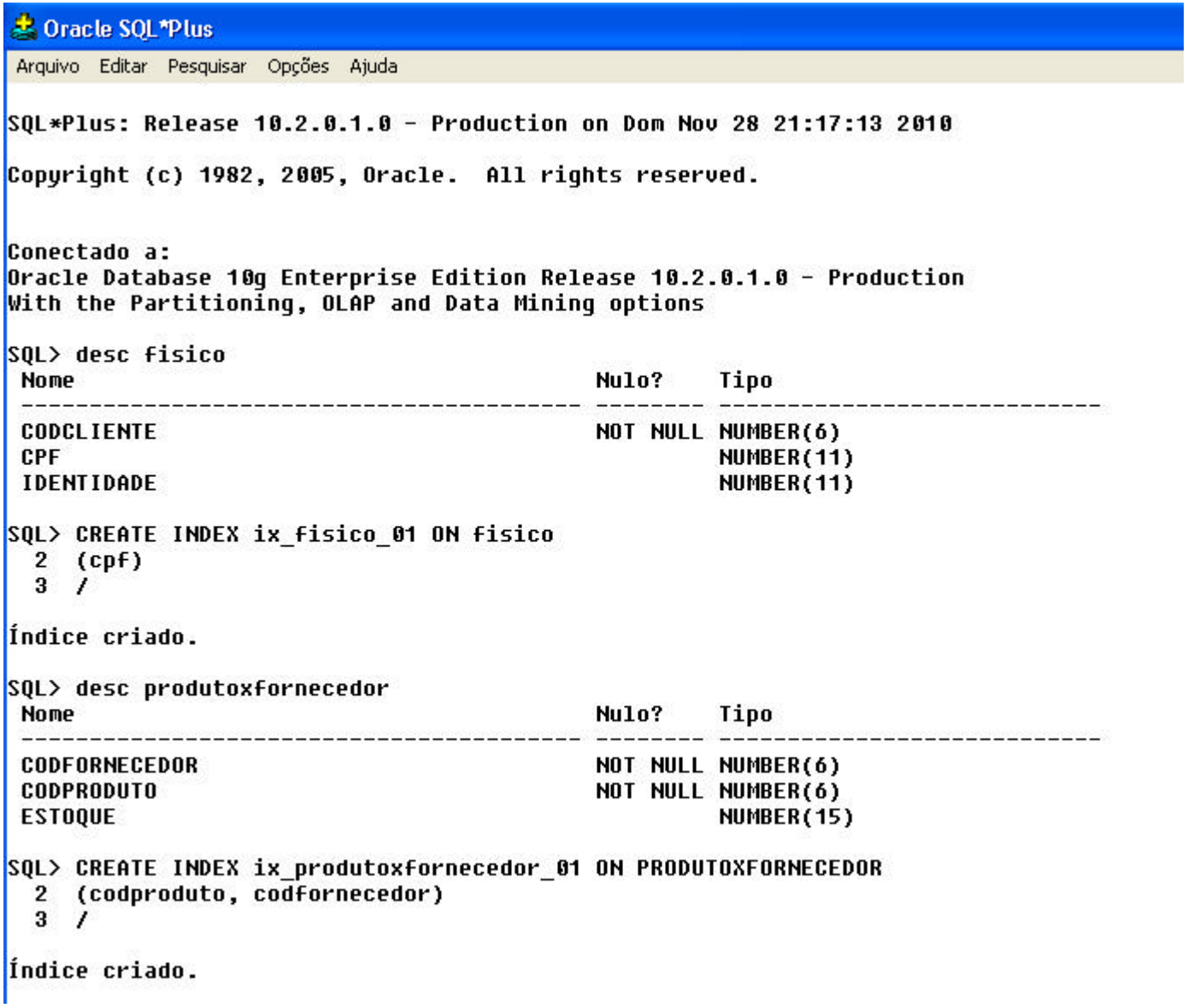
- índice: Nome do índice;
- tabela: Nome da tabela;
- coluna(s): Coluna ou colunas que serão indexadas.

Sintaxe para remoção:

DROP INDEX índice;

- índice: Nome do índice.

8. Criação de Índices no Oracle



9. Uso de Constraints e Índices

Existem dois tipos de constraints que automaticamente criam índices associados. São elas:

- Chave primária (primary key);
- Unique (única).

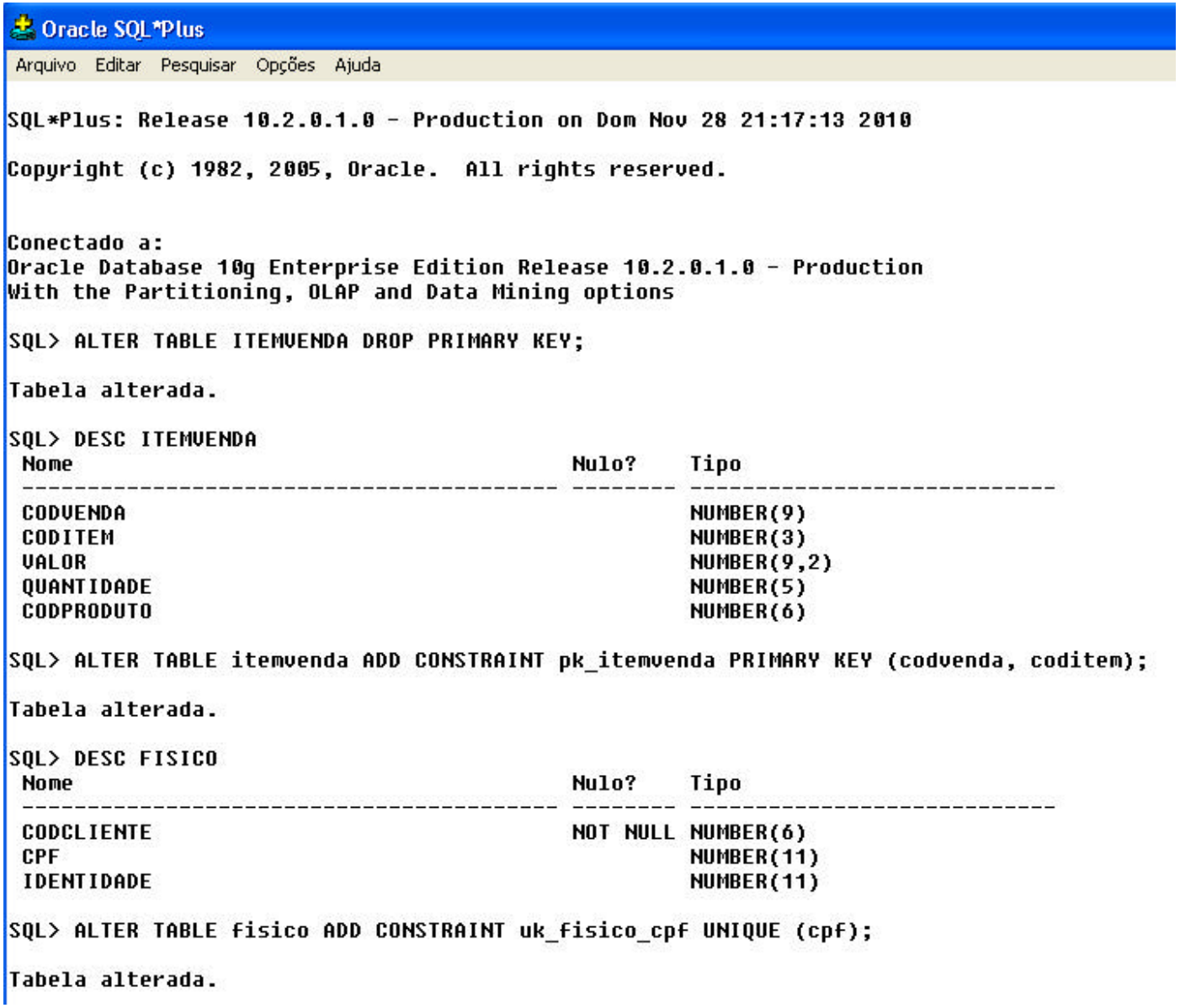
Sintaxe para criação:

ALTER TABLE tabela ADD CONSTRAINT nome_constraint PRIMARY KEY/UNIQUE (coluna[, coluna]...);

Sintaxe para remoção:

Chave primária: **ALTER TABLE tabela DROP PRIMARY KEY;**
Unique: **ALTER TABLE tabela DROP CONSTRAINT nome_constraint;**

9. Uso de Constraints e Índices



10. Checklist para Criação de Índices

Num projeto de banco de dados alguns cuidados devem ser seguidos:

1) Toda tabela deve possuir uma chave primária;

2) Devem ser criados índices para suporte das chaves estrangeiras (foreign keys);

3) Onde ocorrer join entre tabelas, índices devem ser criados para suportar a operação;

4) Devem ser observadas as condições mais freqüêntes de consulta nos programas para criação de índices específicos;

5) Muitos índices em uma tabela podem causar “overhead” em operações de DML.

11. Índices Baseados em Funções

A partir do Oracle versão 10 é possível criar índices no banco de dados baseado em funções, para agilizar consultas como esta:

```
SELECT CODPRODUTO, NOME
FROM PRODUTO
WHERE SUBSTR(NOME,3,1) = 'R';
```

Nestas situações, mesmo que exista um índice pelo campo NOME, ele não será utilizado, fazendo com que o banco de dados tenha que percorrer toda a tabela para encontrar o resultado da consulta. Esta situação é chamada de “FULL TABLE SCAN”.

Obviamente que em tabelas pequenas isso não é problema. Em alguns casos a leitura completa da tabela também pode ser a melhor forma de retornar os dados. Mas pensando em uma consulta que retorna poucos registros de uma tabela muito grande, a opção por um índice para acessá-la seria mais viável.

11. Índices Baseados em Funções