16

管理索引

ORACLE

目标

完成这一课的学习后，您应该能达到下列目标：

- 列出各种类型的索引及其用途
- 创建各种类型的索引
- 重新组织索引
- 维护索引
- 监视索引的使用

ORACLE

索引分类

索引是一种允许对表中的行进行直接访问的树型结构。可以根据索引的逻辑设计或物理实现对索引进行分类。逻辑分类从应用程序的角度对索引进行分组，而物理分类则是基于索引的存储方式。

单列索引和串接索引：

单列索引 在索引键中仅有一列，例如，雇员表中雇员编号列上的索引。

串接索引 是在表内多个列上创建的，也称为“组合索引”。串接索引中的列不必与表中的列顺序一致，也不必相互邻接，例如，雇员表中部门和职务列上的索引。

组合键索引最多包含 32 列。不过，所有列大小的总和不能超过数据块中可用数据空间的大约二分之一（减去某些开销）。

索引分类（续）

唯一索引和非唯一索引：

索引可以是唯一的或非唯一的。唯一索引保证表中没有两行在键列（或多个键列）中具有重复的值。非唯一索引对列值没有此限制。

基于函数的索引：

如果在表中要建立索引的一列或多列上使用了函数或表达式，则创建的是基于函数的索引。基于函数的索引预先计算函数或表达式的值，并将结果存储在索引中。可以将基于函数的索引创建为 B 树或位图索引。

域索引：

域索引是特定于应用程序 (Text, Spatial) 的索引，由索引类型提供的例行程序创建、管理和访问。它之所以称为 “域索引”，是因为它对特定于应用程序的域中的数据建立索引。

仅支持单列域索引。可以在具有标量、对象或 LOB 数据类型的列上建立单列域索引。

分区索引和非分区索引：

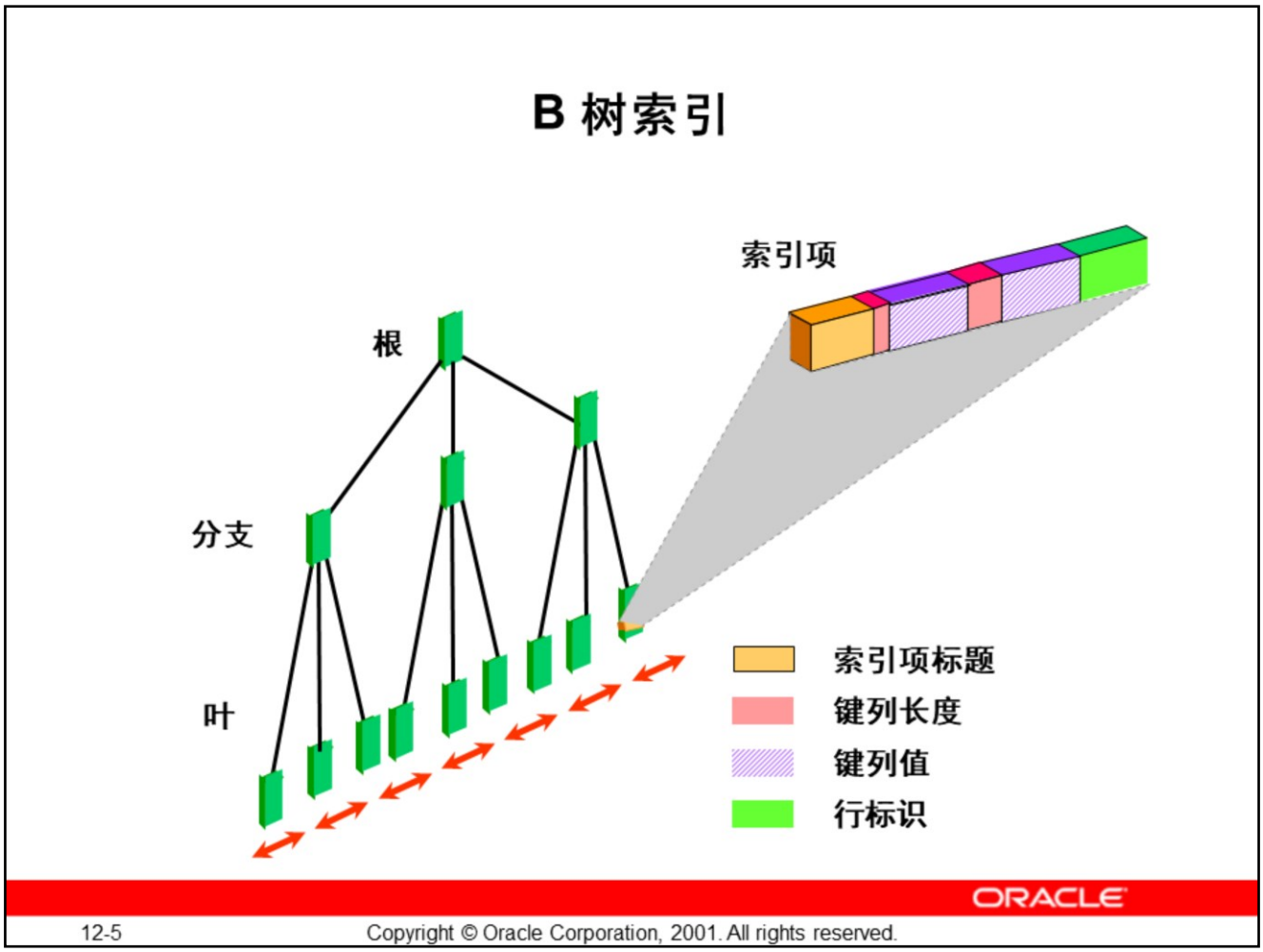
分区索引用于大型表，将与某个索引对应的索引项存储在多个段中。分区使索引可以在多个表空间中展开，从而降低索引查找争用并提高可管理性。分区索引通常用于分区表以提高可伸缩性和可管理性。可为每个表分区创建一个索引分区。

反向键：对索引值反转，第1位与第n位对调，第2位与第n-1位对调，依此类推

目的：解决叶子块的争用问题

优点：消除插入操作的索引热点。对于插入的性能提升有帮助

缺点：不能使用索引范围扫描



B 树索引

虽然所有索引都使用 B 树结构，但术语 “B 树索引” 通常与存储每个关键字的行标识列表的索引关联。

B 树索引的结构：

索引的顶部为根，其中包含指向索引中下一级的项。下一级为分支块，分支块又指向索引中下一级的块。最低一级为叶节点，其中包含指向表行的索引项。叶块为双重链接，有助于按键值的升序和降序扫描索引。

索引叶项的格式：

索引项由以下部分组成：

- 项标题，存储列数和锁定信息
- 键列的 “长度 – 值” 对，用于定义键列的大小及该列的值（该值对的数目即索引中的最大列数。）
- 行的行标识，包含键值

B 树索引（续）

索引叶项的特征：

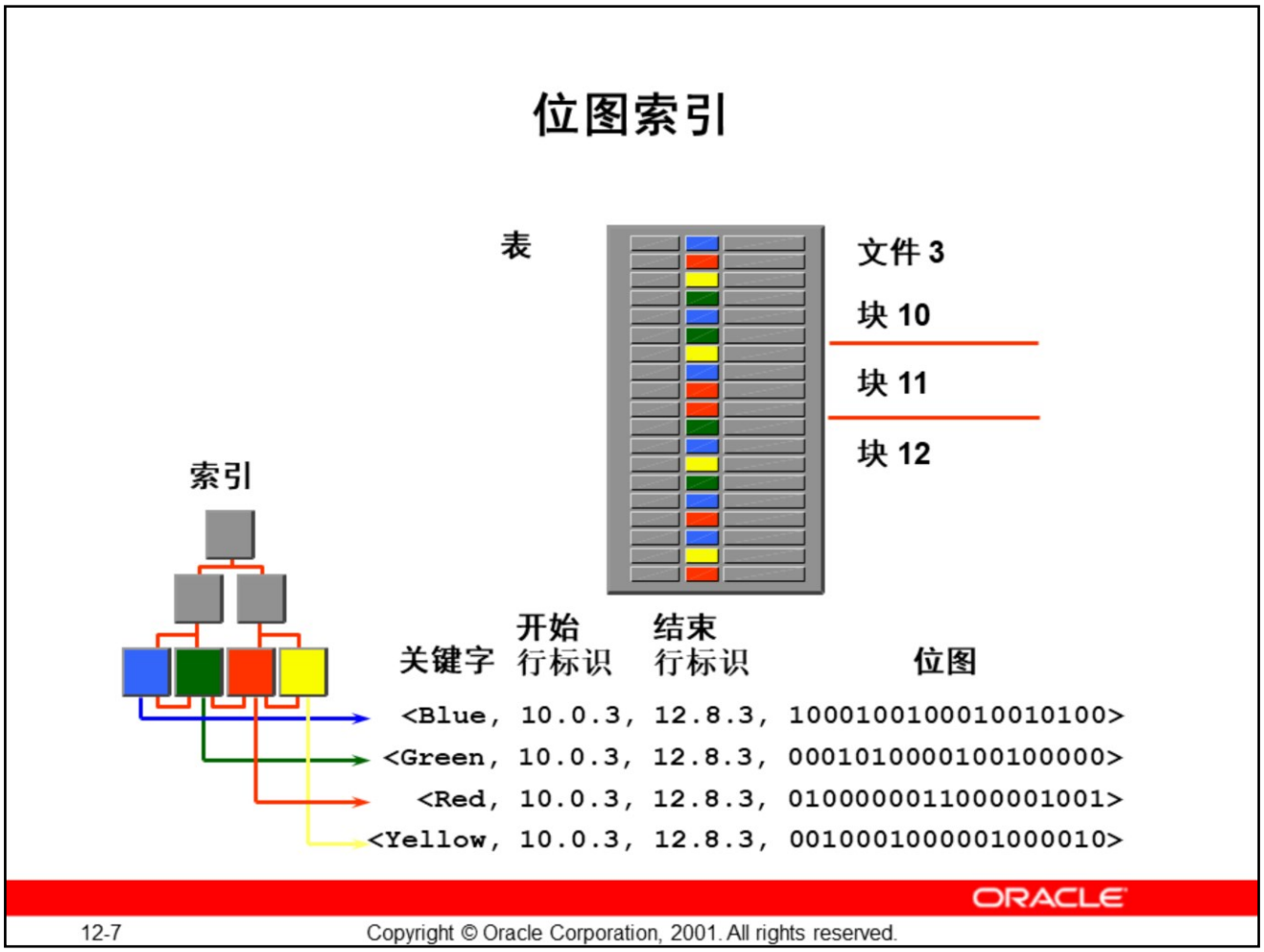
在非分区表上的 B 树索引中：

- 如果多行具有相同的键值，除非对索引进行了压缩，否则键值重复。
- 对于所有键列都为 NULL 的行，没有对应的索引项。因此，指定 NULL 的 WHERE 子句始终进行全表扫描。
- 因为所有行都属于同一段，所以使用受限行标识指向表中的行。

DML 操作对索引的影响：

在表上执行 DML 操作时，Oracle 服务器将维护所有的索引。下面解释 DML 命令对索引的影响：

- 插入操作导致在适当的块中插入索引项。
- 删除行只导致逻辑删除索引项。删除的行所用的空间仍不能用于新项，直到删除块中的所有项。
- 更新键列将导致逻辑删除和向索引插入项。除了创建时以外，PCTFREE 设置在其它任何时候都对索引没有影响。即使索引块空间少于 PCTFREE 指定的空间，仍可以向索引块添加新项。



位图索引

在下列情况中，位图索引比 B 树索引更有利：

- 当表包含数百万行且键列的基数很低（即，该列中重复的值很多）时。例如，对于包含护照记录的表的性别列和婚姻状况列而言，位图索引比 B 树索引更适合
- 当查询经常使用涉及 OR 运算符的多个 WHERE 条件组合时
- 当键列上存在只读或很少的更新操作时

位图索引的结构：

也可以将位图索引组织为 B 树，但叶节点存储每个键值的位图而非行标识列表。位图中的每一位对应一个可能的行标识，如果设置了位，则意味着具有相应行标识的行包含键值。

如图所示，位图索引的叶节点包含下列几项：

- 项标题，包含列数和锁定信息
- 键值由每个键列的 “长度 – 值” 双值组成（本例中，关键字仅包含一列，第一项的键值为 Blue）。

位图索引（续）

位图索引的结构（续）：

- 开始行标识，本例中的开始行标识包含文件号 3、块号 10 和行号 0
- 结束行标识，本例中的结束行标识包含块号 12 和行号 8
- 位图段，由位串组成（对应的行包含键值时设置位；不包含键值时不设置位。Oracle 服务器使用专利压缩技术存储位图段。）

开始行标识是位图的位图段指向的第一行的行标识，即，位图的第一位对应此行标识，位图的第二位对应块中的下一行，而结束行标识是位图段所包含的表中最后一行的指针。位图索引使用受限行标识。

使用位图索引：

B 树用于定位包含给定键值的位图段的叶节点。开始行标识和位图段用于定位包含键值的行。

更改表中的键列时，必须修改位图。这将导致锁定相关的位图段。由于锁是在整个位图段上获取的，所以直到第一个事务处理结束后，才能由其它事务处理更新位图包含的行。

比较 B 树索引和位图索引

B 树	位图
适用于高基数列	适用于低基数列
更新关键字的成本相对较低	更新键列的成本非常高
使用 OR 谓词进行查询时效率较低	使用 OR 谓词进行查询时效率较高
对 OLTP 很有用	对数据仓库很有用

ORACLE

比较 B 树索引和位图索引

用于低基数列时，位图索引比 B 树索引更紧凑。

由于位图使用位图段级锁定，所以位图索引中的键列的更新成本较高；而在 B 树索引中，锁位于与表中单个行相对应的项上。

位图索引可用于执行位图布尔等操作。Oracle 服务器可以使用两个位图段执行逐位布尔操作并得到一个结果位图。这将允许在使用布尔谓词的查询中更有效地使用位图。

总之，B 树索引更适合索引动态表的 OLTP 环境，而位图索引更适合在大型静态表上使用复杂查询的数据仓库环境。

创建正常的 B 树索引

索引可在表所有者的帐户下或其它帐户下创建，但索引通常在表所在的同一帐户下创建。

上面的语句在 EMPLOYEES 表上创建一个索引（使用 LAST_NAME 列）。

创建正常的 B 树索引（续）

语法选项

UNIQUE：用于指定唯一的索引（缺省为 Nonunique。）

Schema：索引/表的所有者

Index：索引名

Table：表名

Column：列名

ASC/DESC：指示是按升序还是按降序创建索引

TABLESPACE：指定要在其中创建索引的表空间

PCTFREE：创建索引时为容纳新的索引项而在每块中保留的空间大小（以总空间量减去块头后的百分比表示）

INITRANS：指定每块中预先分配的事务处理项的数目（缺省值和最小值为 2。）

MAXTRANS：限制可以为每个块分配的事务处理项数（缺省值为 255。）

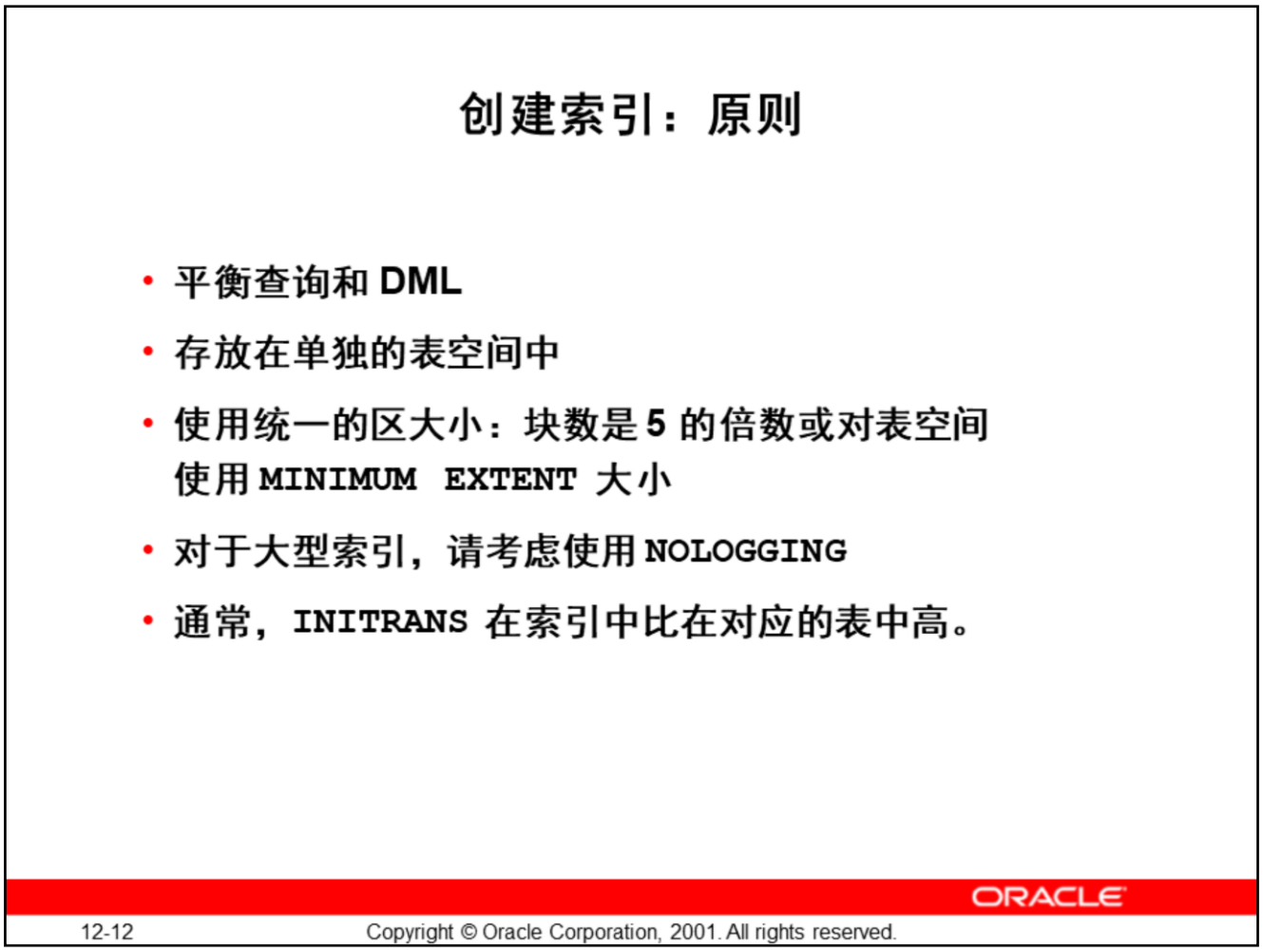
LOGGING：指定在重做日志文件中记录索引创建操作和在索引上执行的后续操作（这是缺省值。）

NOLOGGING：指定在重做日志文件中不记录创建操作和某些类型的数据加载操作

NOSORT：指定将行按升序存储在数据库中，这样，Oracle 服务器在创建索引时不必对行进行排序

注

- 如果已为表空间定义了 MINIMUM EXTENT，则索引的区大小将向上舍入为下一个更高的 MINIMUM EXTENT 值的倍数。
- 如果省略了 [NO]LOGGING 子句，索引的事件记录属性将缺省为表所驻留的表空间的事件记录属性。
- 不能为索引指定 PCTUSED。由于索引项必须按正确的顺序存储，所以用户无法控制何时在某一索引块中插入。
- 如果在数据未按关键字排序的情况下使用 NOSORT 关键字，语句将终止并显示错误。如果表上已经有多个 DML 操作，则该选项很可能无效。
- 如果可能，Oracle 服务器使用现有索引创建新的索引。当新索引的关键字与现有索引键的主要部分对应时，就会发生这种情况。



创建索引：原则

创建索引时应考虑：

- 索引能够提高查询性能并降低 DML 操作速度。始终使易失表所需的索引数保持最少。
- 将索引放在一个单独的表空间中，不要放在有还原段、临时段和表的表空间中。
- 对大型索引而言，避免生成重做日志可显著提高性能。请考虑使用 NOLOGGING 子句创建大型索引。
- 由于索引项比索引行小，所以索引块趋向于在每块中包含更多的项。因此，INITRANS 在索引中通常比在对应的表中高。

索引和 PCTFREE：

索引的 PCTFREE 参数与表的 PCTFREE 参数工作方式不同。前者仅在创建索引时用来为需要插入到同一索引块的索引项保留空间。而不更新索引项。更新键列时，这将涉及逻辑删除索引项和插入。

创建索引：原则（续）

索引和 PCTFREE（续）

在单调递增（如系统生成的发票号）列的索引上使用较低的 PCTFREE 值。在这些情况下，新的索引项总是追加到现有项上，没有必要在两个现有索引项间插入一个新项。

如果插入行的索引列值可采用任何值（即新值在当前的值范围内），则应该提供较高的 PCTFREE。发票表的客户代码列上的索引就是一个要求高 PCTFREE 值的索引。在这种情况下，将 PCTFREE 值指定为由下列等式所表示的值是非常有用的：

$$\frac{\text{最大行数} - \text{初始行数}}{\text{最大行数}} \times 100$$

最大值可用于特定的时间周期，如一年。

创建位图索引

语法：

使用下列命令创建位图索引：

```
CREATE BITMAP INDEX [schema.] index
ON [schema.] table
(column [ ASC | DESC ] [ , column [ASC | DESC ] ] ...)
[ TABLESPACE tablespace ]
    [ PCTFREE integer ]
    [ INITRANS integer ]
    [ MAXTRANS integer ]
    [ LOGGING| NOLOGGING ]
    [ NOSORT ]
```

注意，位图索引不能是唯一的。

CREATE_BITMAP_AREA_SIZE 参数：

初始化参数 CREATE_BITMAP_AREA_SIZE 决定了内存中用于存储位图段的空间量。缺省

值为 8 MB。使用较大的值，可提高索引创建的速度。如果基数很小，可将该值设置为一个较小值。例如，如果基数仅为 2，则该值可以为千字节数量级而非兆字节数量级。一般来讲，基数越大，则获取最佳性能所需的内存越多。

创建簇索引

```
CREATE INDEX ind_test
ON cluster clu_test;
```

ORACLE

创建分区索引

- 独立分区（只能是B树索引）
  - 范围分区
  - 哈希分区
- 与表分区相同

ORACLE

范围分区索引

```
Create index ind_emp1_salary on emp1(salary)
Global partition by range(salary)
(partition par_2000 values less than (2000) tablespace test1,
Partition par_4000 values less than(4000) tablespace test2,
Partition par_other values less than (maxvalue) tablespace test3)
;
```

哈希分区索引

```
Create index ind_emp2_name on emp2(name)
Global partition by hash(name)
(partition par_1 tablespace test1,
Partition par_2 tablespace test2,
Partition par_3 tablespace test3)
;
```

与表分区相同分区的索引

CREATE bitmap INDEX ind_emp3_job
ON emp3(job) local;

查询分区索引信息

User_part_indexes：分区类别、数量描述

User_part_key_columns：分区字段描述

User_ind_partitions：每个分区的细节描述

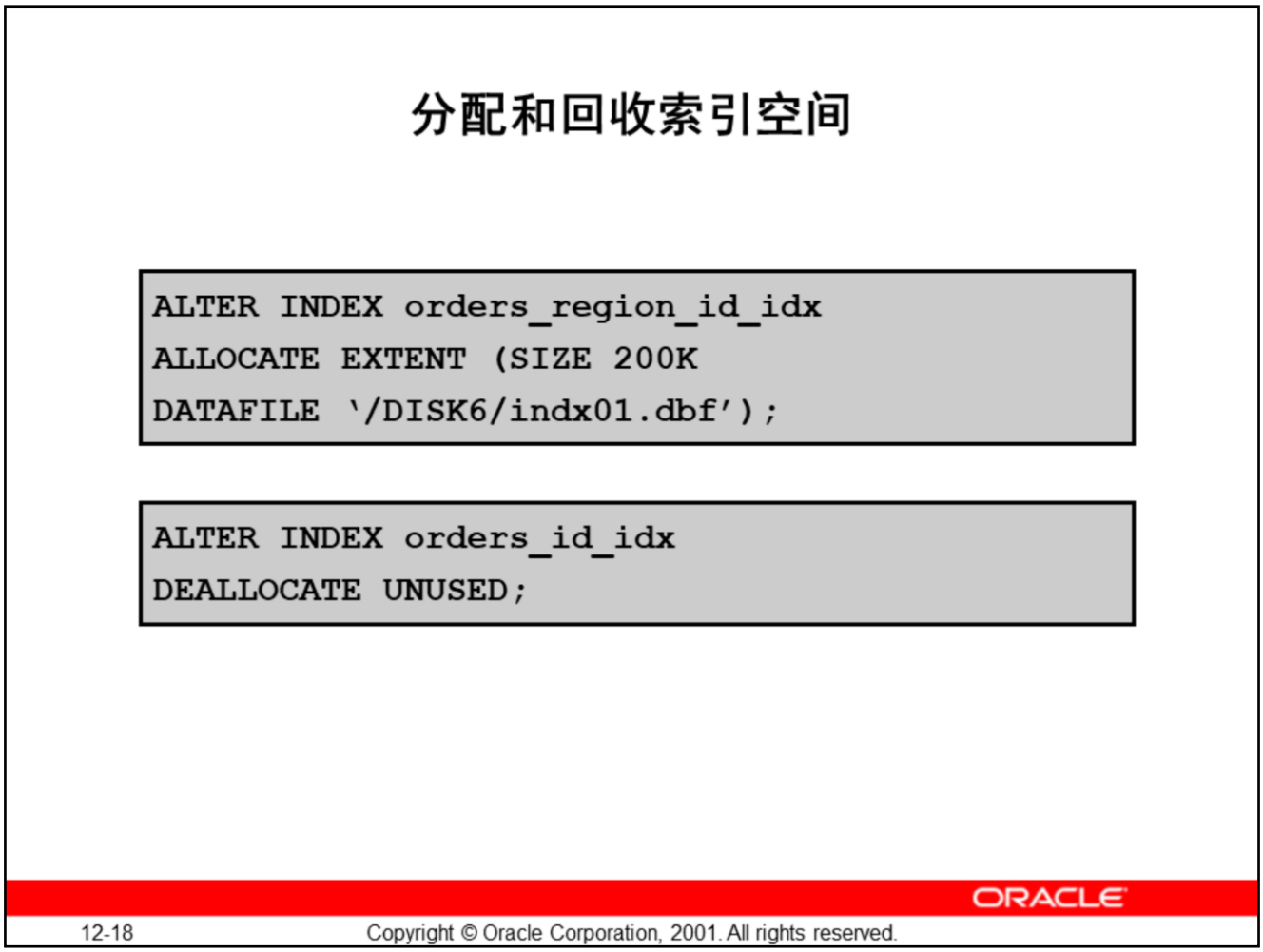
更改索引的参数

块使用参数可通过 ALTER INDEX 命令进行修改。

语法：

```
ALTER INDEX [schema.]index
[ INITRANS integer ]
[ MAXTRANS integer ]
```

可更改块使用参数以保证在索引块上实现更高级别的并发性。



分配和回收索引空间

手动分配索引空间：

在表上进行频繁的插入操作前，可能需要向索引添加区。添加区可防止索引动态扩展并导致性能降低。

从索引中手动回收空间：

使用 ALTER INDEX 命令的 DEALLOCATE 子句释放索引中超过高水位标记的未用空间。

语法：

使用下列命令分配或回收索引空间：

```
ALTER INDEX [schema.]index
{ALLOCATE EXTENT ([SIZE integer [K|M]]
                  [ DATAFILE 'filename' ])
| DEALLOCATE UNUSED [KEEP integer [ K|M ] ] }
```

手动分配和手动回收索引空间所遵循的规则与在表上使用这些命令时相同。

注：当建立索引的表被截断时，回收索引空间。截断表将导致截断关联的索引。

重建索引

索引重建具有以下特点：

- 将现有索引作为数据源建立新索引。
- 使用现有索引建立索引时无需排序，从而使性能更佳。
- 在建立新索引后，删除旧索引。在重建期间，各自的表空间内需要有足够的空间以容纳新旧索引。
- 结果索引不包括任何已删除的项。因此，该索引可以更有效地使用空间。
- 在建立新索引的过程中，查询可继续使用现有索引。

可能的重建情况：

在下列情况下应重建索引：

- 需要将现有索引移到另外的表空间中。如果索引和表在同一表空间中或者需要跨磁盘重新分布对象时，可能需要执行此操作。

重建索引（续）

可能的重建情况（续）：

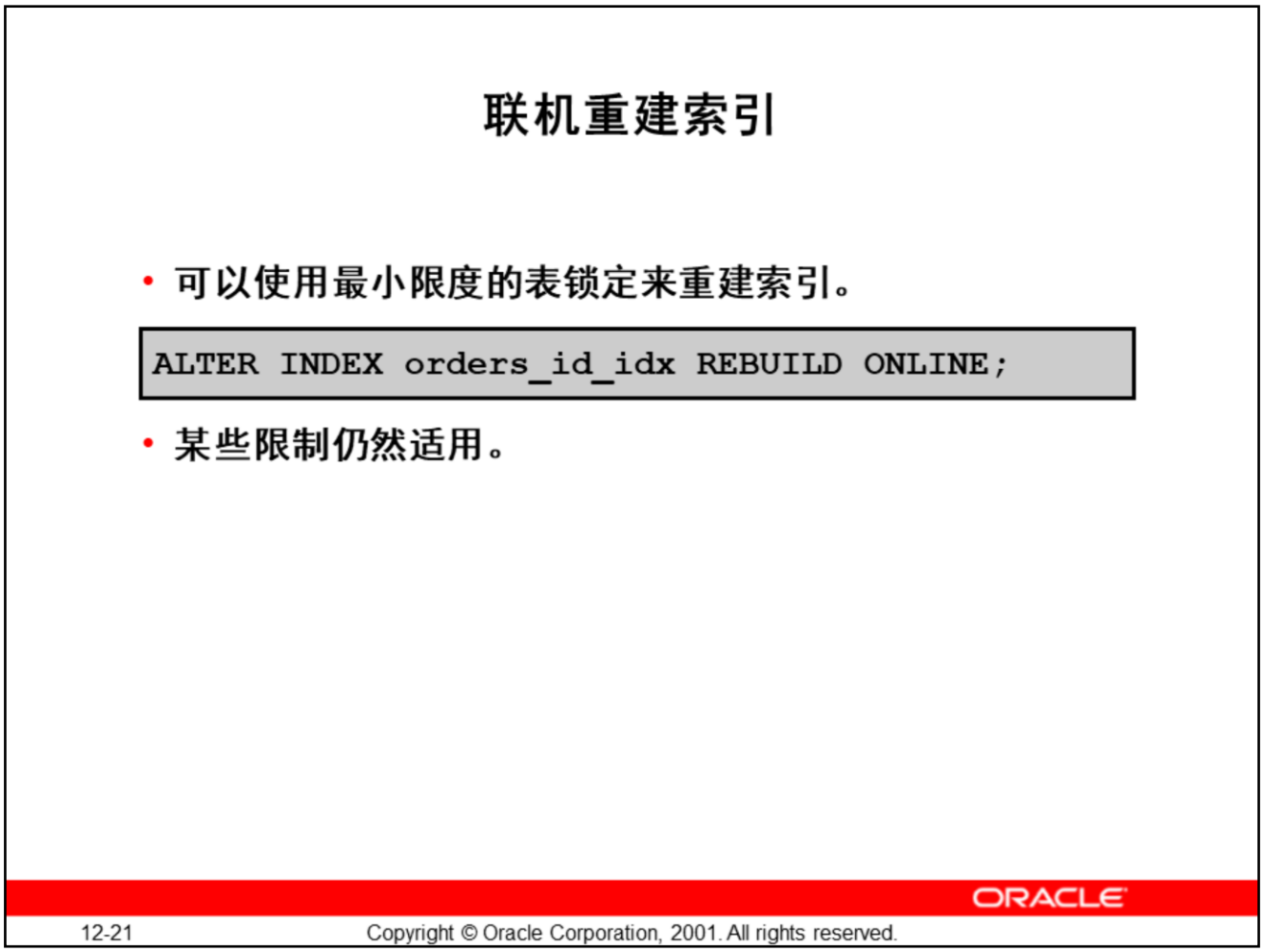
- 索引中包含很多已删除的项。这是滑动索引（如订单表中的订单号上的索引）存在的典型问题，完成的订单已被删除，并将具有更高订单号的新订单添加到表中。如果有几个旧订单未完成，则可能有若干个索引叶块包含除几个已删除项之外的全部项。
- 需要将现有正常索引转换成反向键索引。在从 Oracle 服务器的早期发行版移植应用程序时，可能会出现这种情况。
- 已通过 ALTER TABLE...MOVE TABLESPACE 命令将索引表移至其它表空间。

语法：

使用下列命令重建索引：

```
ALTER INDEX [schema.] index REBUILD
[ TABLESPACE tablespace ]
[ PCTFREE integer ]
[ INITRANS integer ]
[ MAXTRANS integer ]
[ LOGGING| NOLOGGING ]
[ REVERSE | NOREVERSE ]
```

ALTER INDEX ...REBUILD 命令不能用于将位图索引更改为 B 树索引，反之亦然。只能为 B 树索引指定 REVERSE 或 NOREVERSE 关键字。



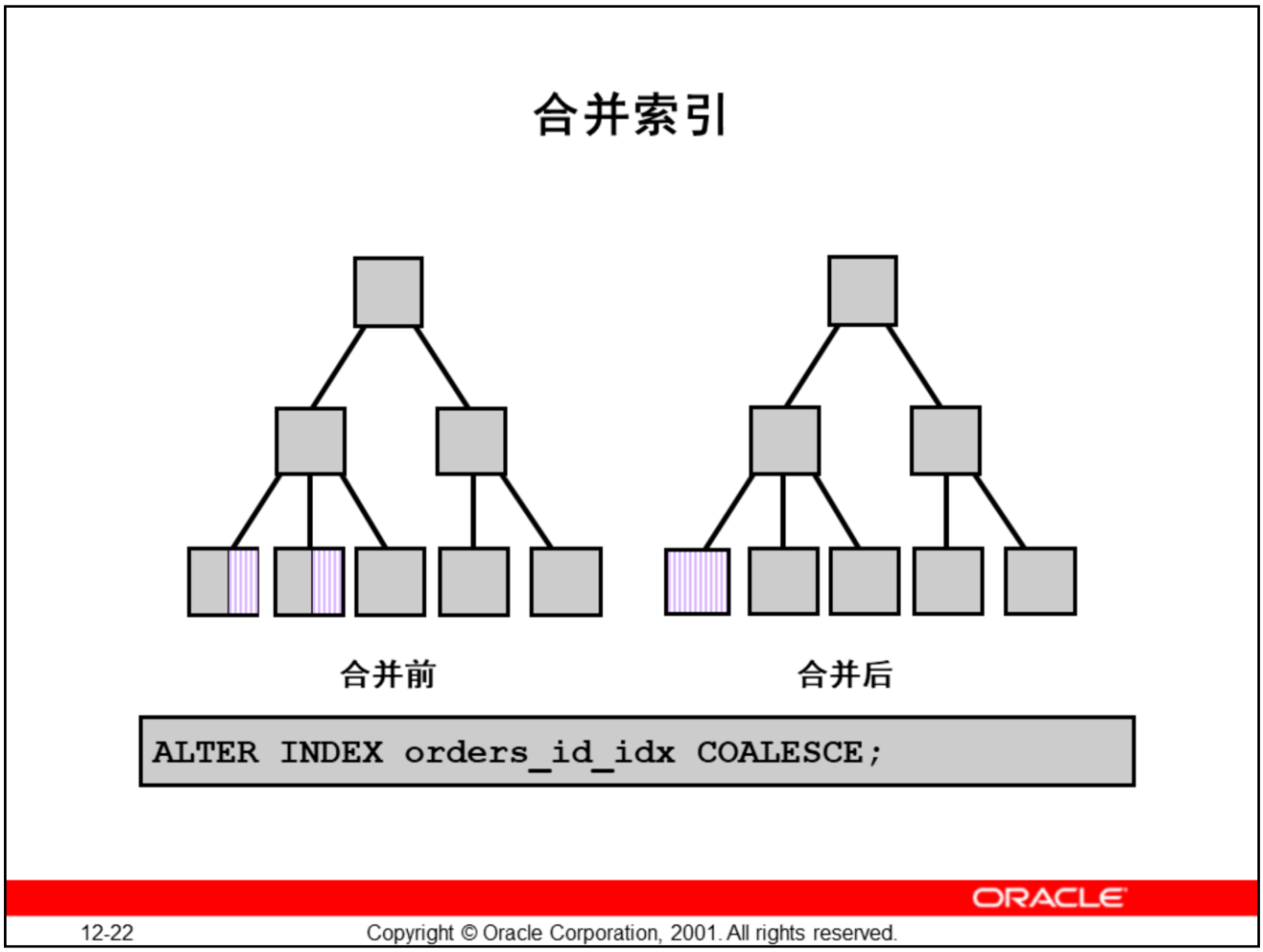
联机重建索引

建立或重建索引是一项费时的任务，尤其当表非常大时更是如此。在 Oracle早期版本，建立或重建索引都需要锁定表，并要防止并发的 DML 操作。 Oracle 允许在基表上进行并发操作的同时建立或重建索引，但不建议在此过程中执行大量的 DML 操作。

注：仍存在 DML 锁，这意味着在联机索引建立期间不能执行其它 DDL 操作。

限制：

- 不能在临时表中重建索引
- 不能重建整个分区索引。必须分别重建每个分区或子分区。
- 也不能回收未用空间。
- 不能整个更改索引的 PCTFREE 参数值。



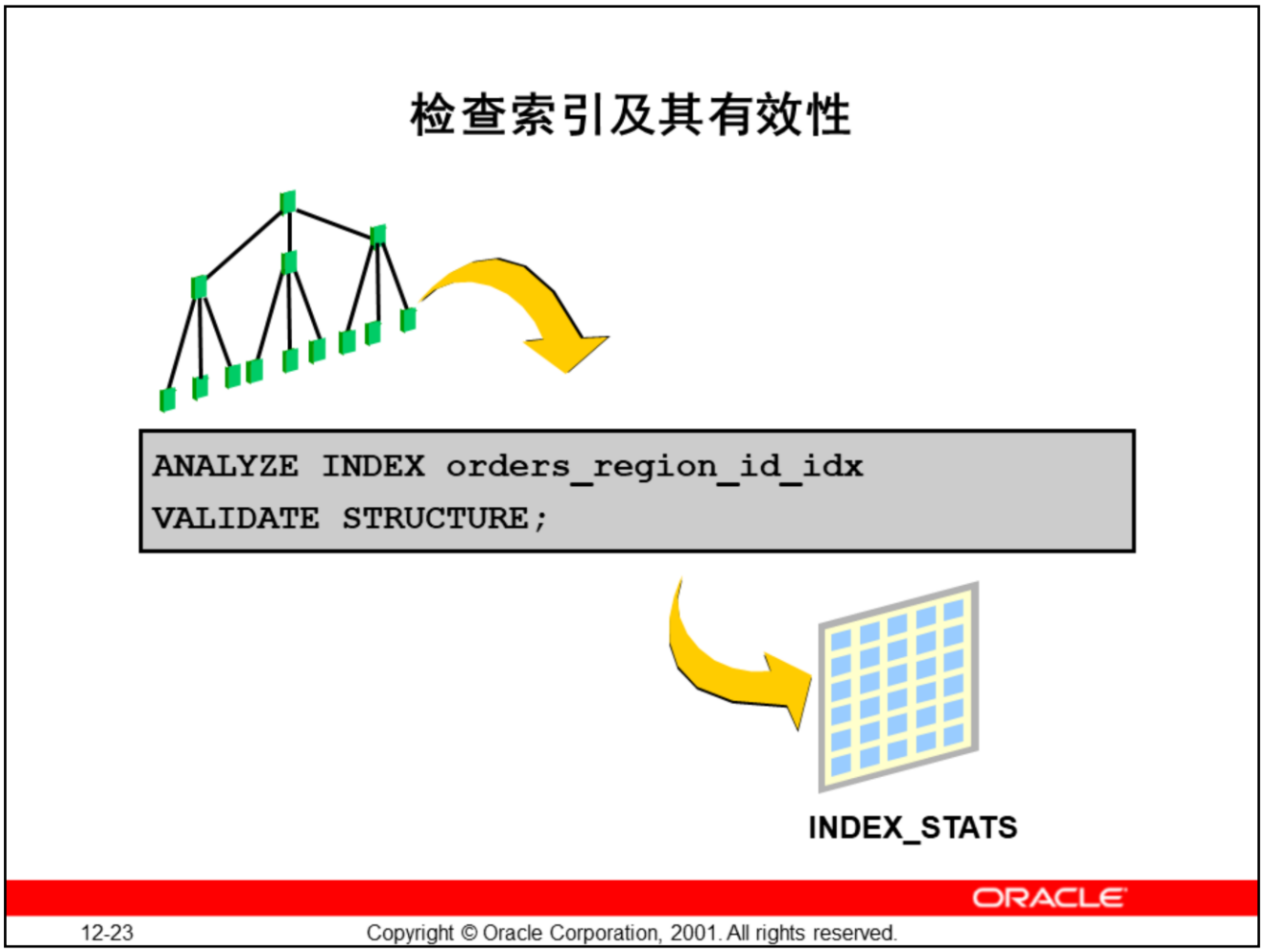
合并索引

遇到索引碎片时，可以重建或合并索引。执行上述任务前，应考虑每种选择的成本和好处，然后选择最适合自己情况的方案。索引合并是联机完成的块重建过程。

如果有可以释放以供重用的 B 树索引叶块，则可使用下列 SQL 语句合并这些叶块：

```
SQL> ALTER INDEX hr.employees_idx COALESCE;
```

上图显示了 ALTER INDEX … COALESCE 语句对索引 hr.employees_idx 的影响。在执行 COALESCE 操作之前，前两个叶块为 50% 填满。这意味着，索引包含碎片，可以对索引进行合并以完全填充第一个块，从而减少了碎片。



检查索引及其有效性

分析索引以执行以下操作：

- 检查所有的索引块是否存在损坏。注意，此命令并不验证索引项是否与表中的数据对应。
- 使用索引的有关信息填充 INDEX_STATS 视图。

语法：

```
ANALYZE INDEX [ schema.]index VALIDATE STRUCTURE
```

运行此命令后，查询 INDEX_STATS 以获取索引的有关信息，如下例所示：

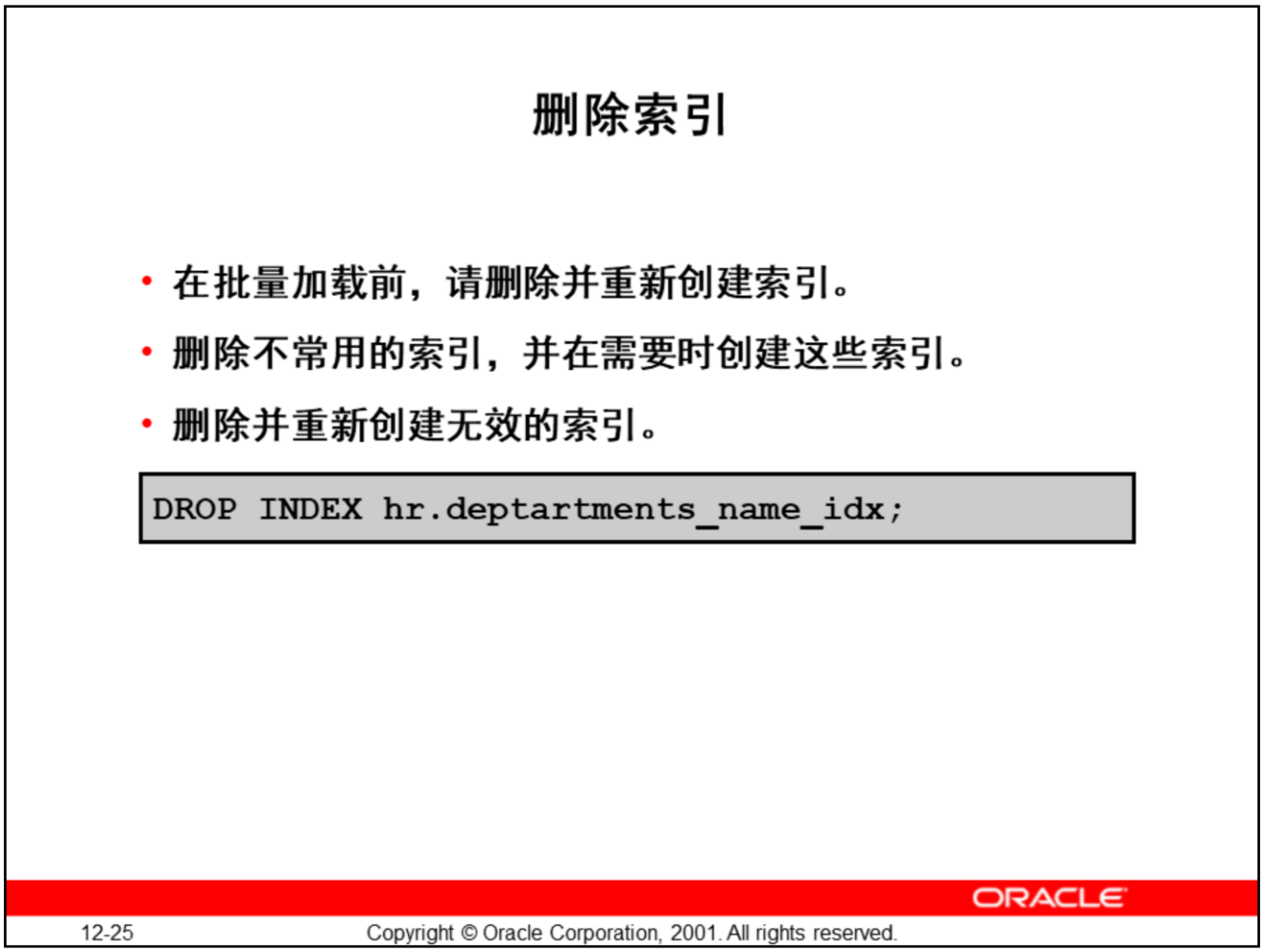
检查索引及其有效性（续）

```
SQL>  SELECT blocks, pct_used, distinct_keys
  2   lf_rows, del_lf_rows
  3   FROM index_stats;

    BLOCKS     PCT_USED   LF_ROWS    DEL_LF_ROWS
    ------     ---------  --------   ------------
    25         11         14         0

  1 row selected.
```

如果索引中已删除行的比例很高，请重新组织该索引。例如：当 DEL_LF_ROWS 占 LF_ROWS 的比率超过 30% 时。



删除索引

下列情况应删除索引：

- 应用程序不再需要索引时，可将索引删除。
- 执行批量加载前，索引可能已删除。在大量加载数据前，先删除索引，加载后再重新创建索引，这样做的好处有：
  - 提高加载性能
  - 更有效地使用索引空间
- 仅定期使用的索引无需不必要的维护，尤其在基于易失表时更是如此。这是 OLTP 系统中的通常情况，在该系统中，年末或季度末会生成特殊的查询，以收集在总结会上使用的信息。
- 当在某种类型的操作（如加载）期间出现例程失败时，可能会将索引标记为 `INVALID`。在这种情况下，需要删除并重建索引。
- 索引已损坏。

不能删除约束所需的索引，因此，必须先禁用或删除相关的约束。

标识未用索引

- 要开始监视索引的使用，请执行以下语句：

```
ALTER INDEX hr.dept_id_idx
MONITORING USAGE
```

- 要停止监视索引的使用，请执行以下语句：

```
ALTER INDEX hr.dept_id_idx
NOMONITORING USAGE
```

ORACLE

标识未用索引

可以在 V$OBJECT_USAGE 中收集和显示有关索引使用的统计信息。如果收集的信息表明索引从未使用过，则删除该索引。此外，删除未用索引还可减少 Oracle 服务器用于 DML 操作的开销，从而改善了性能。每次指定 MONITORING USAGE 子句时，将对指定的索引重置 V$OBJECT_USAGE。以前的信息被清除或重置，并记录新的开始时间。

V$OBJECT_USAGE 列

INDEX_NAME：索引名

TABLE_NAME：对应的表

MONITORING：指示监视是 ON 还是 OFF

USED：指示 YES 或 NO，即在监视时间内是否使用了索引

START_MONITORING：索引监视的开始时间

END_MONITORING：索引监视的结束时间

获取索引信息

可以通过查询以下视图来获取有关索引的信息：

- DBA_INDEXES：提供有关索引的信息
- DBA_IND_COLUMNS：提供有关索引列的信息
- V$OBJECT_USAGE：提供有关索引使用情况的信息

ORACLE

小结

在这一课中，您应该能够掌握：

- 创建各种类型的索引
- 重新组织索引
- 删除索引
- 从数据字典获取索引信息
- 启动和结束对索引使用情况的监视

ORACLE