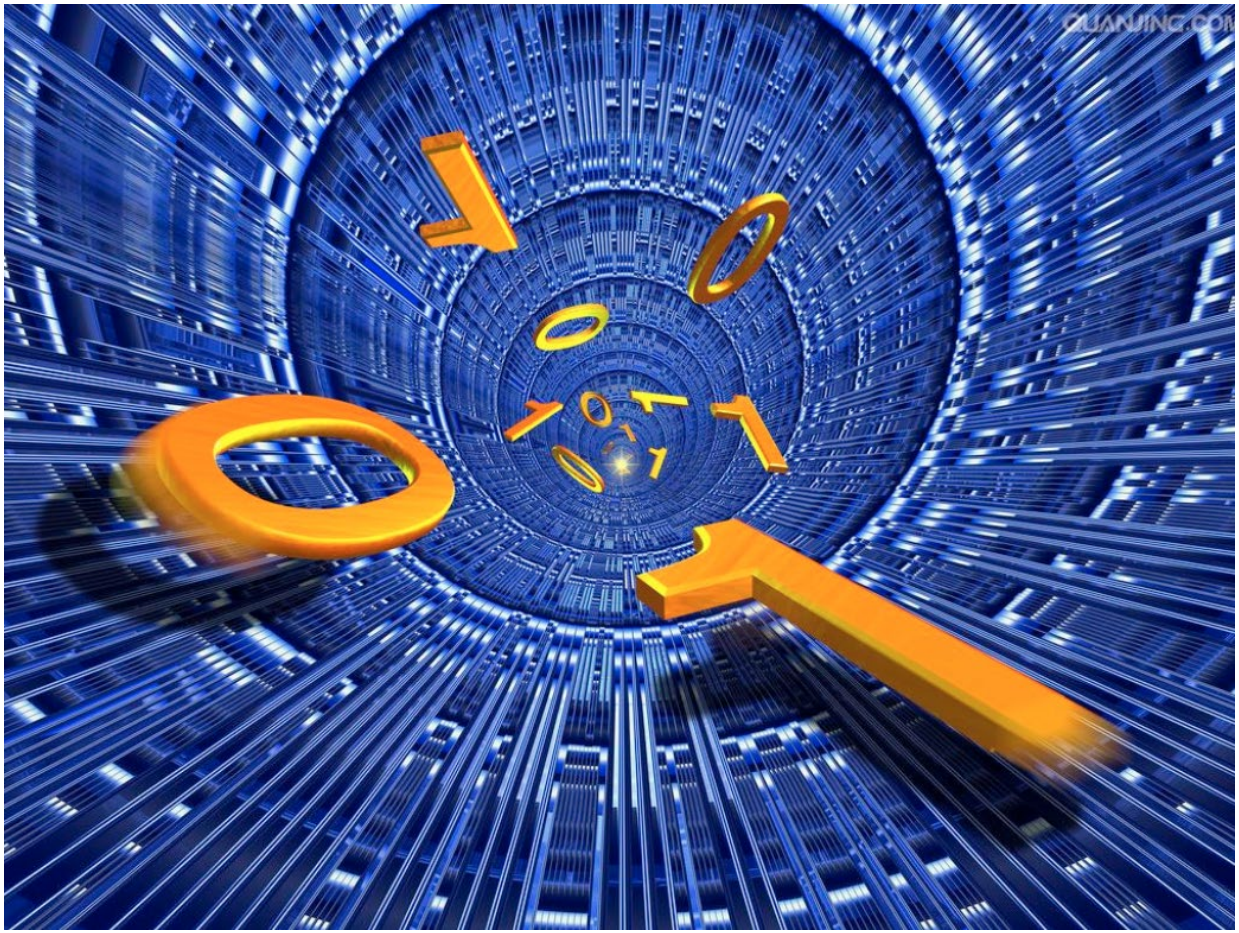
中山大學
SUN YAT-SEN UNIVERSITY



# 2017级《数据库原理与应用》第4周

2020.5.29

中山大学数学学院 黄志洪

# 综合练习

- 列出工资比最高工资的一半要高的员工
- 列出平均工资超过公司平均工资的部门名称
- 列出人数最多的两个工种
- 列出人数最多的部门名称
- 列出工资差（最高-最低）最大的部门名称

# 创建用户

```
SQL> create user y1 identified by abc;

用户已创建。

SQL> create user y2 identified by 123;
create user y2 identified by 123
                             *
第 1 行出现错误:
ORA-00988: 口令缺失或无效


SQL> create user y2 identified by "123";

用户已创建。

SQL>
```

# 更改口令

```
SQL> alter user y1 identified by xyz
  2  ;

用户已更改。

SQL>
```

# 删除用户

- Drop user 用户名;
- Drop user 用户名 cascade;

# 用户权限

- 系统特权：支配系统中一般性资源的能力，一般由DBA授予
- 对象特权：支配某一具体数据库对象的能力，一般由对象拥有者授予

# 查看所有系统特权

```
SQL> connect  / as sysdba
已连接。
SQL> desc dba_sys_privs
 名称                                      是否为空? 类型
 ----------------------------------------- -------- ----------------------------
 GRANTEE                                   NOT NULL VARCHAR2(30)
 PRIVILEGE                                 NOT NULL VARCHAR2(40)
 ADMIN_OPTION                                       VARCHAR2(3)

SQL> select privilege from dba_sys_privs where grantee='DBA';

PRIVILEGE
----------------------------------------
CHANGE NOTIFICATION
ADMINISTER ANY SQL TUNING SET
ALTER ANY SQL PROFILE
CREATE RULE
EXPORT FULL DATABASE
EXECUTE ANY EVALUATION CONTEXT
DEQUEUE ANY QUEUE
DROP ANY INDEXTYPE
ALTER ANY INDEXTYPE
EXECUTE ANY LIBRARY
CREATE ANY LIBRARY
```

# 几个有趣的系统特权

- SELECT ANY TABLE
- DROP USER，CREATE USER
- CREATE SESSION
- UNLIMITED TABLESPACE

# 向用户授予系统特权

```
SQL>
SQL> grant create user to y1;

授权成功。

SQL> grant create user to y2 with admin option;

授权成功。
```

# 撤销用户的系统特权

```
SQL> revoke create user from y2;
```

撤销成功。

# 对象特权

```
SQL>
SQL>
SQL> grant connect to y1;

授权成功。

SQL> connect scott/tiger
已连接。
SQL> grant select on emp to y1;

授权成功。

SQL> grant select on dept to y1 with grant option;

授权成功。

SQL>
```

# 比较

- With admin option和with grant option的区别？

# 有哪些对象特权?

- Select
- Update
- Insert
- Delete
- Alter
- Index
- Execute

# 撤销对象特权

```
SQL>
SQL> revoke select on dept from y1;

撤销成功。

SQL>
SQL>
```

# 角色（role）

- 什么是角色——角色是权限和角色的集合
- 角色作用——简化授权管理

# 创建角色

```
SQL> create role r1;
```

角色已创建。

```
SQL> grant select on emp to r1;
```

授权成功。

```
SQL> connect / as sysdba
```
已连接。
```
SQL> grant create user to r1;
```

授权成功。

```
SQL> grant r1 to y1;
```

授权成功。

# 删除角色

- Drop　role　角色名;
- 权限/角色树形结构

# 公众用户public

- 什么是公众用户
- 对公众用户授权的作用

# 同义词

- 什么是同义词?
- 同义词的创建和删除
- 公众同义词

```
SQL> create synonym e1 for scott.emp;

同义词已创建。

SQL> select * from e1;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL
---------- ---------- --------- ---------- -------------- ---------- ---
    DEPTNO
----------
      7369 SMITH      CLERK           7902 17-12月-80             800
        20
```

# 序列

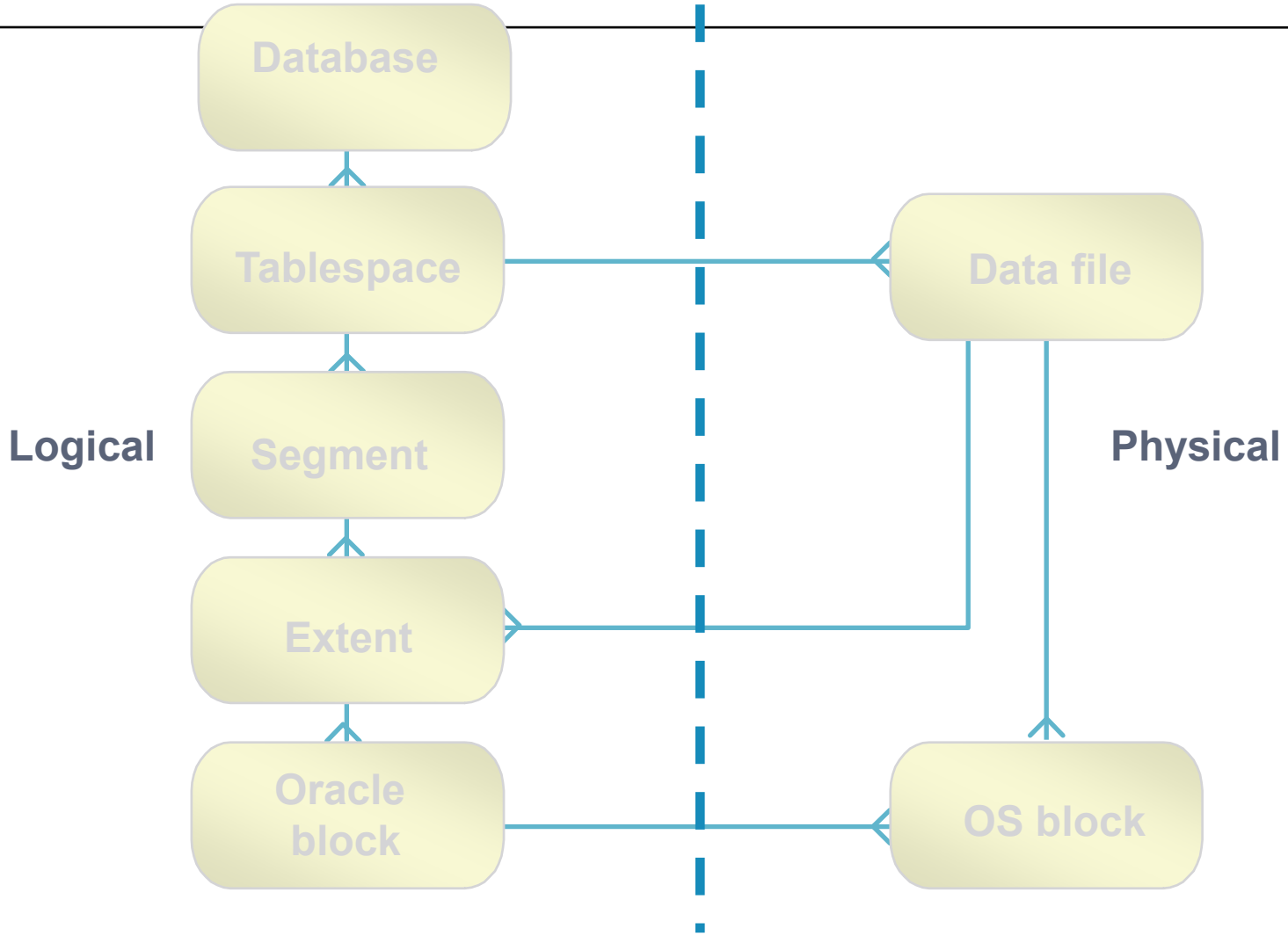
- 什么是序列
- 序列的作用
- 创建序列
- 使用序列

```
SQL> connect scott/tiger
已连接。
SQL>  create sequence a1 start with 1 increment by 10;

序列已创建。

SQL>  select scott.a1.nextval from dual;

   NEXTVAL
----------
         1

SQL> /

   NEXTVAL
----------
        11

SQL> /

   NEXTVAL
----------
        21
```

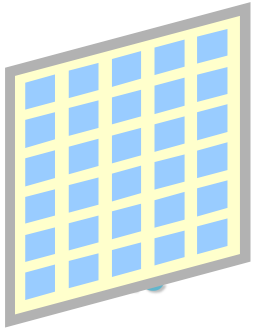
# 索引

- 索引的原理与作用
- 创建和删除索引
- 索引的使用与维护

# 组织层次


Database
Tablespace
Segment
Extent
Oracle block
Data file
OS block
Logical
Physical

# 各种段类型



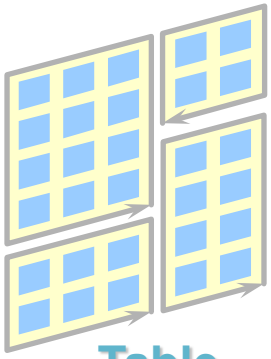
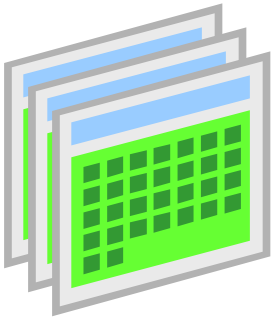
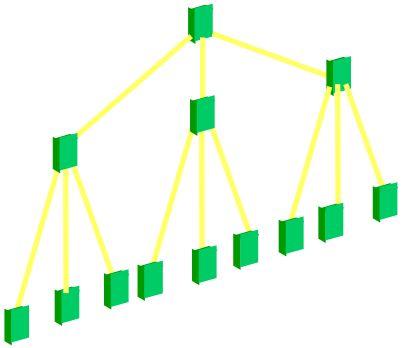


Table partition

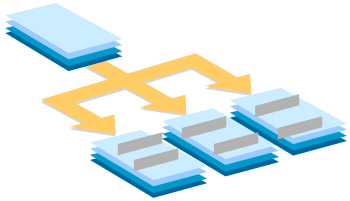


Cluster

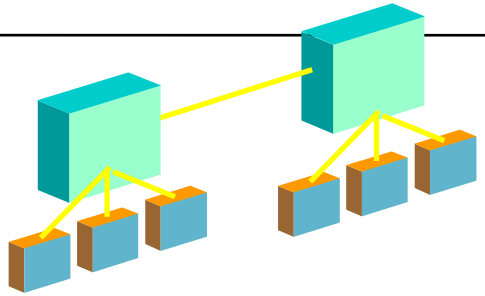


Index

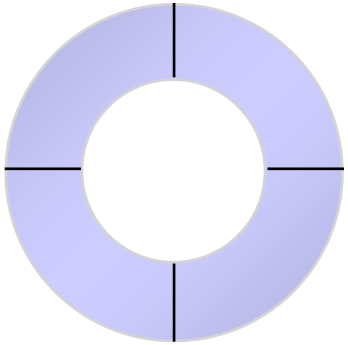
# 各种段类型



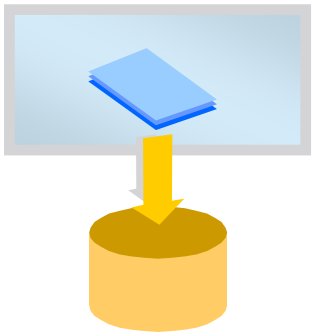
Index-organized table



Index partition



Rollback segment

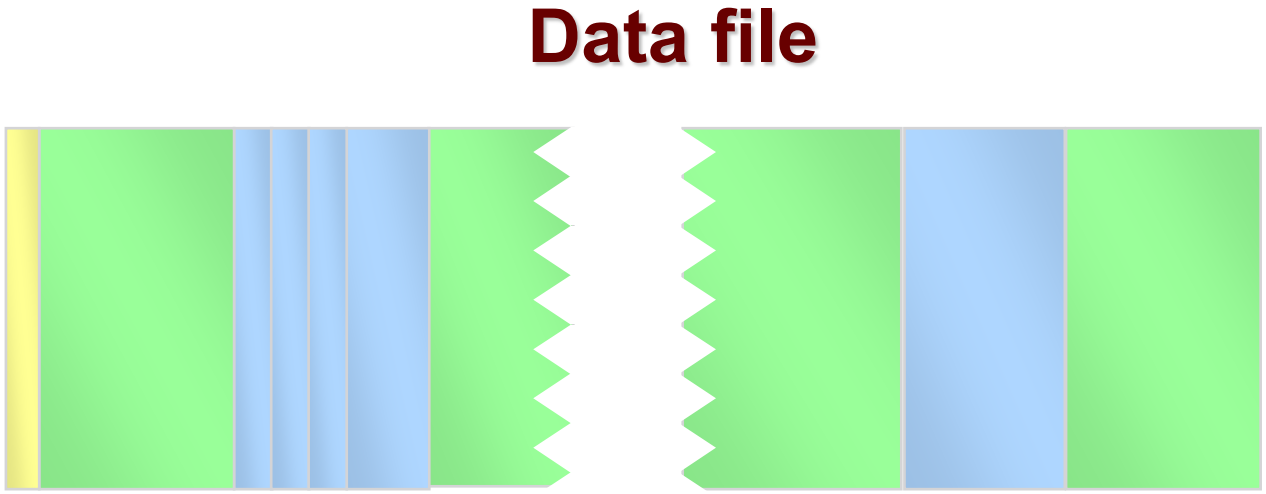


Temporary segment

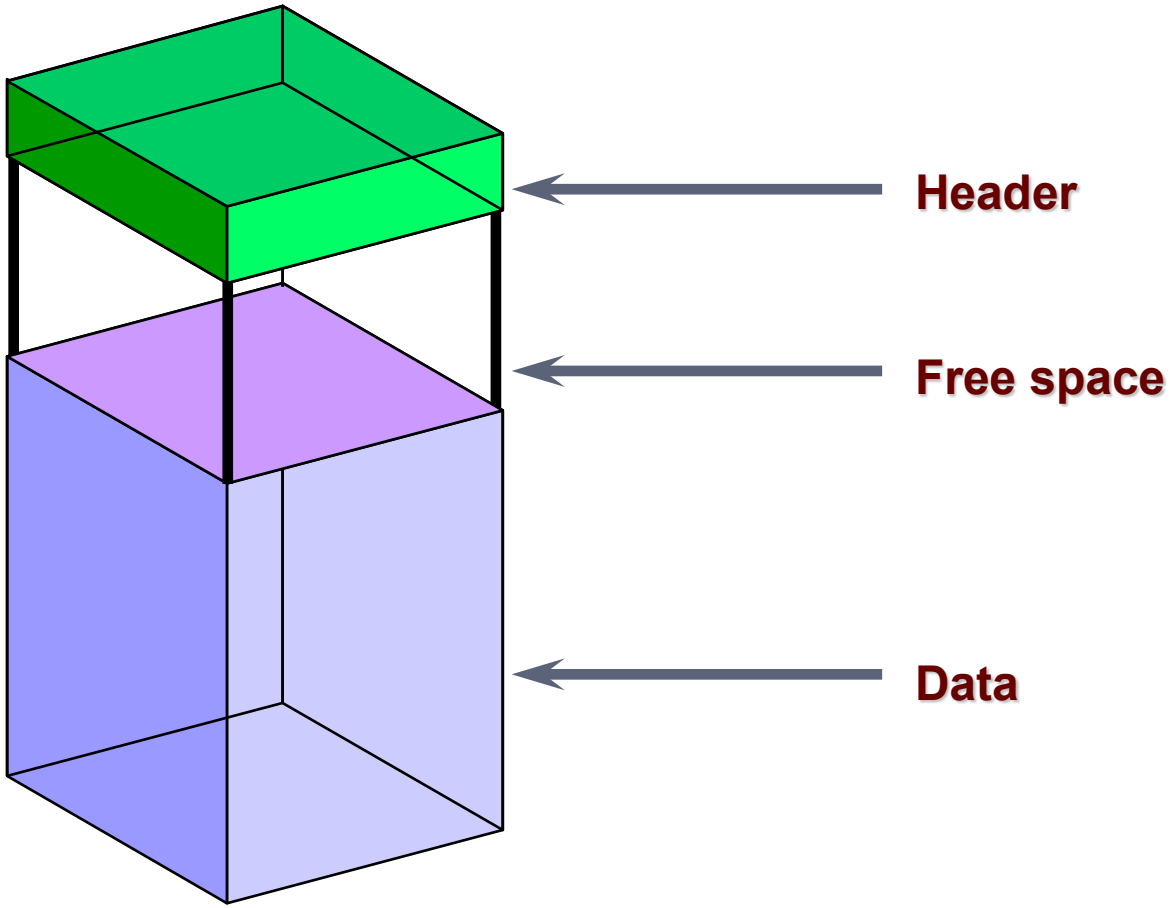
# 范围

- Allocated when the segment is:
  - Created
  - Extended
  - Altered
- Deallocated when the segment is:
  - Dropped
  - Altered
  - Truncated
  - Automatically resized (rollback segments only)

# Used and Free Extents



# 块



# 数据库元素的表示

- 《数据库系统实现》第二章
- 变长元素是个难点
- 核心问题

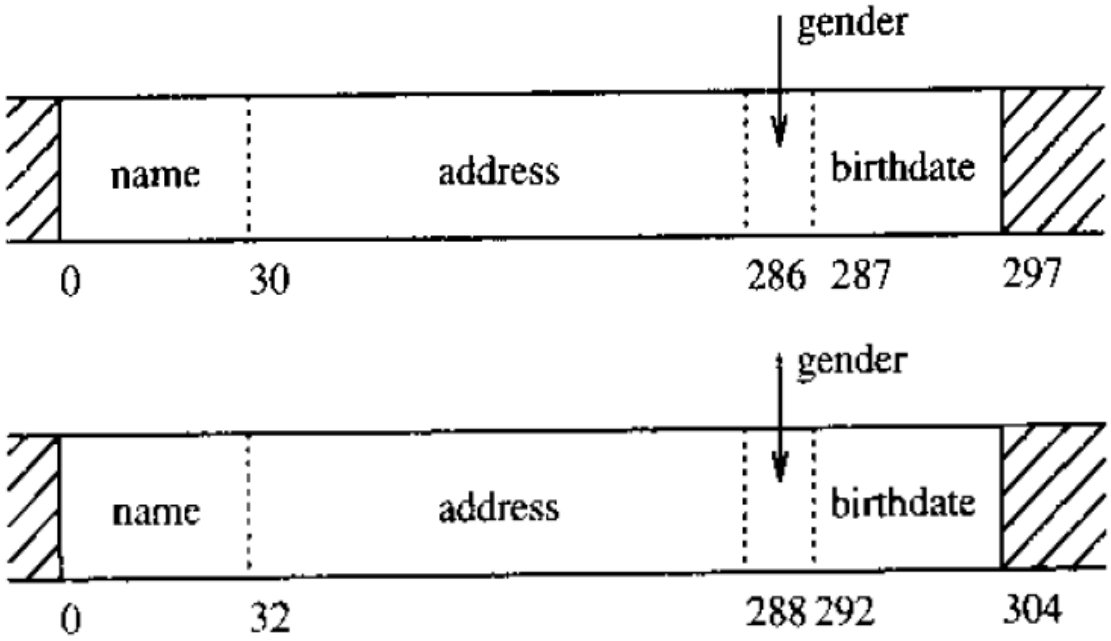
1) 如何将SQL数据类型表示成字段？
2) 如何将元组表示成记录？
3) 如何在存储块中表示记录或元组的集合？
4) 如何用块的集合表示和存储关系？
5) 如果不同的元组可能具有不同的记录大小，或者记录大小不能整除块大小，或二者兼而有之，这时我们如何处理记录大小？
6) 如果因为修改一些字段而导致记录大小发生改变，那么将会发生什么情况？我们如何在记录所在的块内找到存储空间，特别是当记录增大时？

# 数据库元素的表示：字段

- 定长字符串的表示方法
- 变长字符串的表示方法
- 日期和时间类型的表示方法
- 枚举类型的表示

# 字段组装为记录

- 定长记录构造



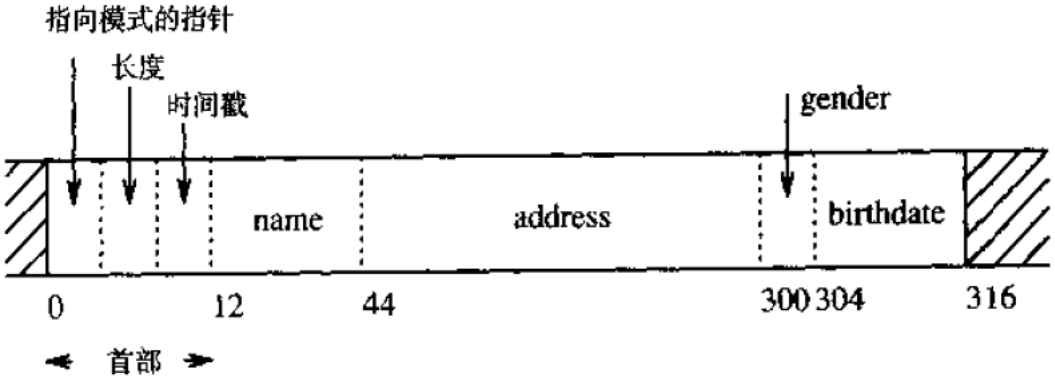
# 记录首部

- 记录首部需要存储的信息

1) 记录模式，或更可能是指向DBMS中存储该类记录模式的位置的一个指针；
2) 记录长度；
3) 时间戳，指明记录最后一次被修改或被读的时间以及其他可能的信息。因此，许多记录格式包括一个由数目不多的字节组成的首部，以提供这种额外信息。

数据库系统维护模式信息，模式信息主要是出现在为那个关系所写的CREATE TABLE 语句中的信息：

1) 关系的属性。
2) 属性类型。
3) 属性在元组中出现的顺序。
4) 属性或关系自身上的约束，如主键声明，或约束某个整数属性的值必须在某一范围内。



# 定长记录在块中的放置

- 块是数据库与磁盘交互的最小单位
- 修改记录必须先读入块，在内存里修改块内容再回写到磁盘

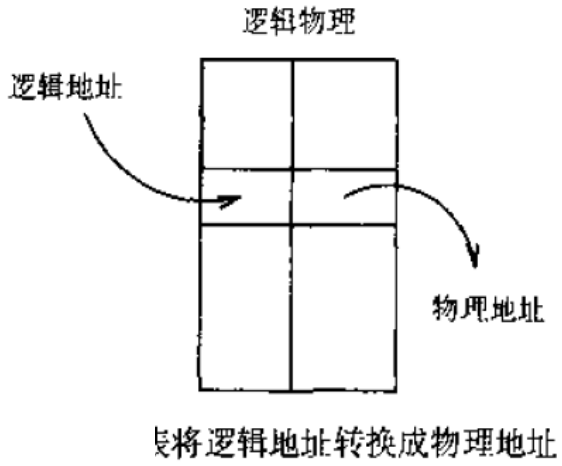

图3-6 一个典型的存储记录的块

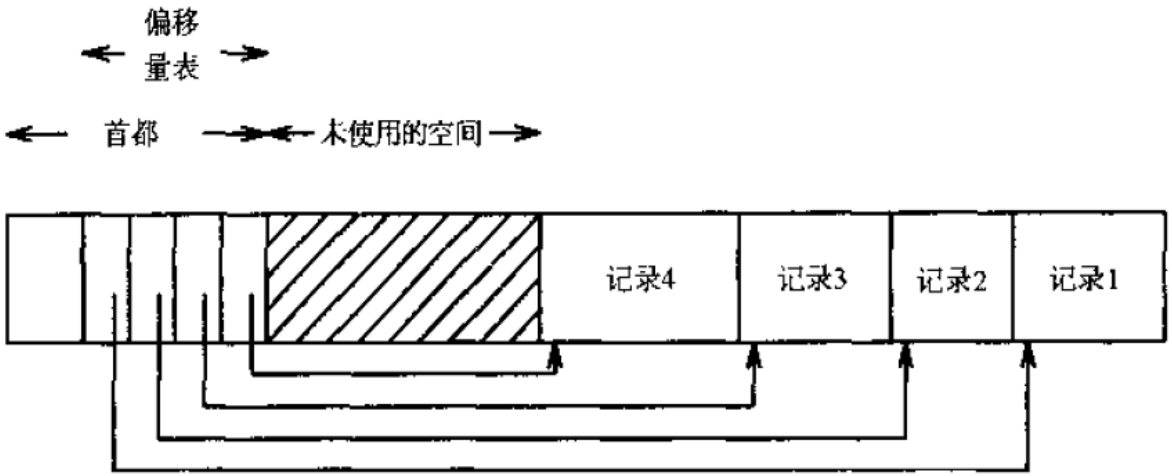
它有一个可选的块首部，存储诸如以下各种信息：

1) 与一个或多个其他块的链接，这些块构成一个块的网络，例如在第4章中所描述的为一个关系的元组创建索引的块。
2) 关于这个块在这样一个网络中所扮演的角色的信息。
3) 关于这个块的元组属于哪个关系的信息。
4) 一个给出每一条记录在块内偏移量的“目录”。
5) 一个“块ID”，参见3.3节。
6) 指明块最后一次修改和/或存取时间的时间戳。

# 块地址与记录（行）地址

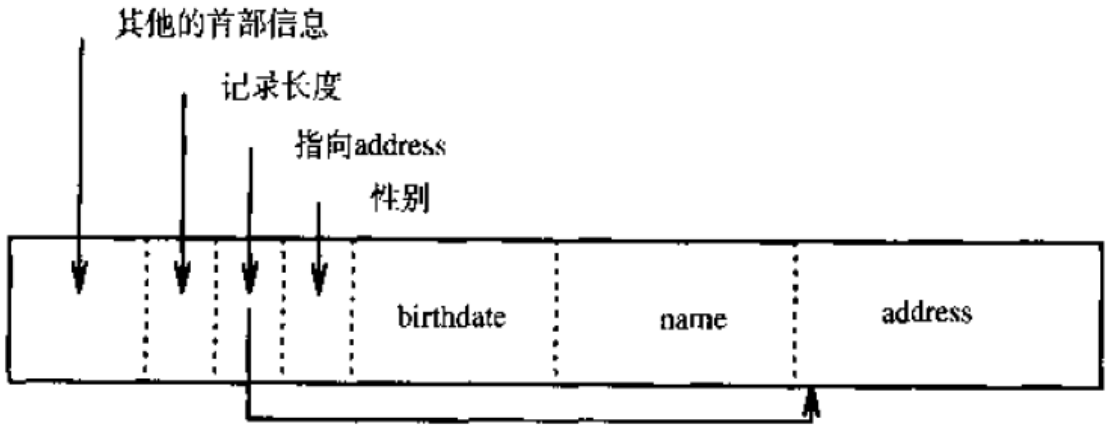
- 逻辑地址和物理地址
- 由数据库/操作系统完成逻辑地址到物理地址的映射
- 怎样通过行地址定位行?
- 运用指针



# 变长字段和记录

- 大量场景拥有变长字段
- 一种组装方法



# 超大记录

- 不能装入一个块的记录，行链
- BLOB类型数据

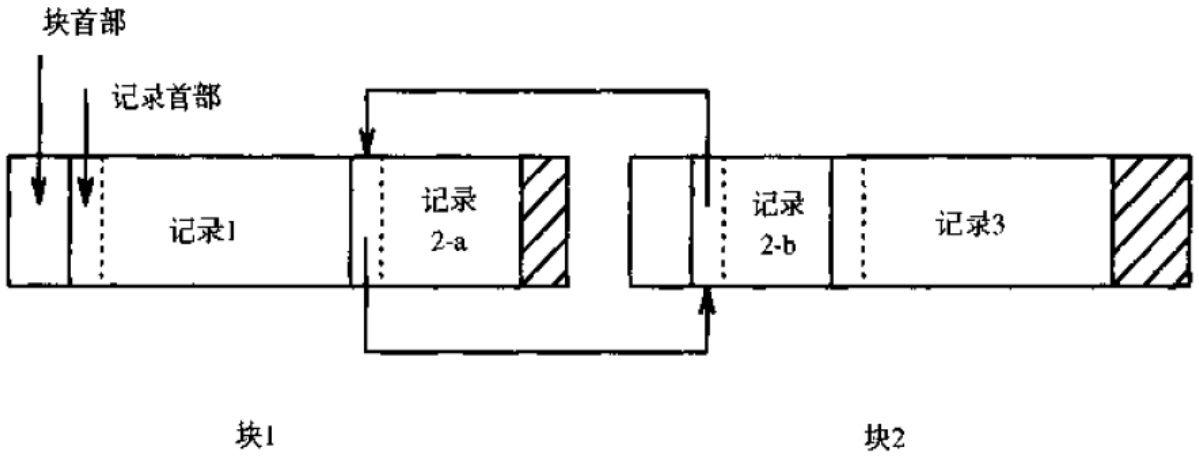



图3-16 跨多个块存储跨块记录

# 插入记录

- 为了寻找合适的插入空间，可能需要再块内“滑动记录”
- 空间溢出的问题（行迁移问题）

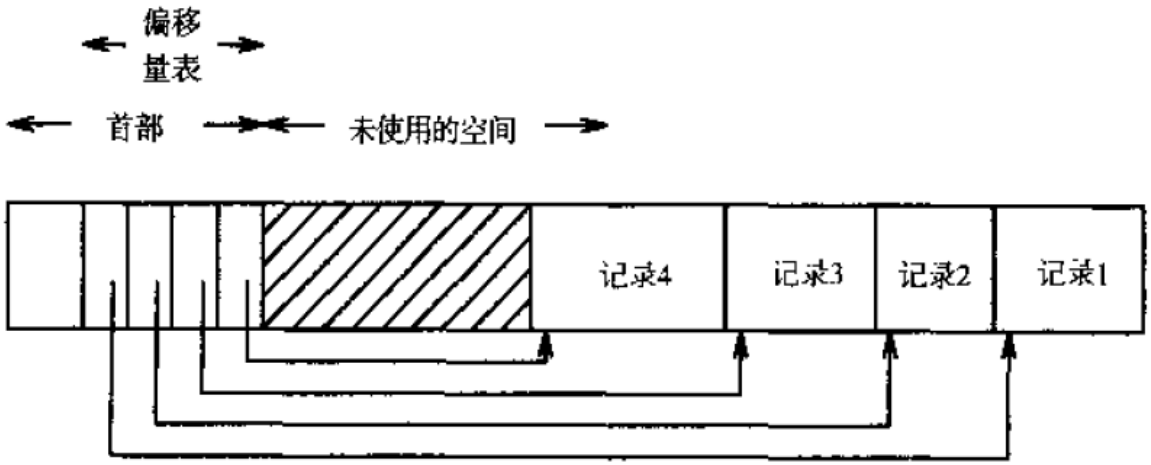
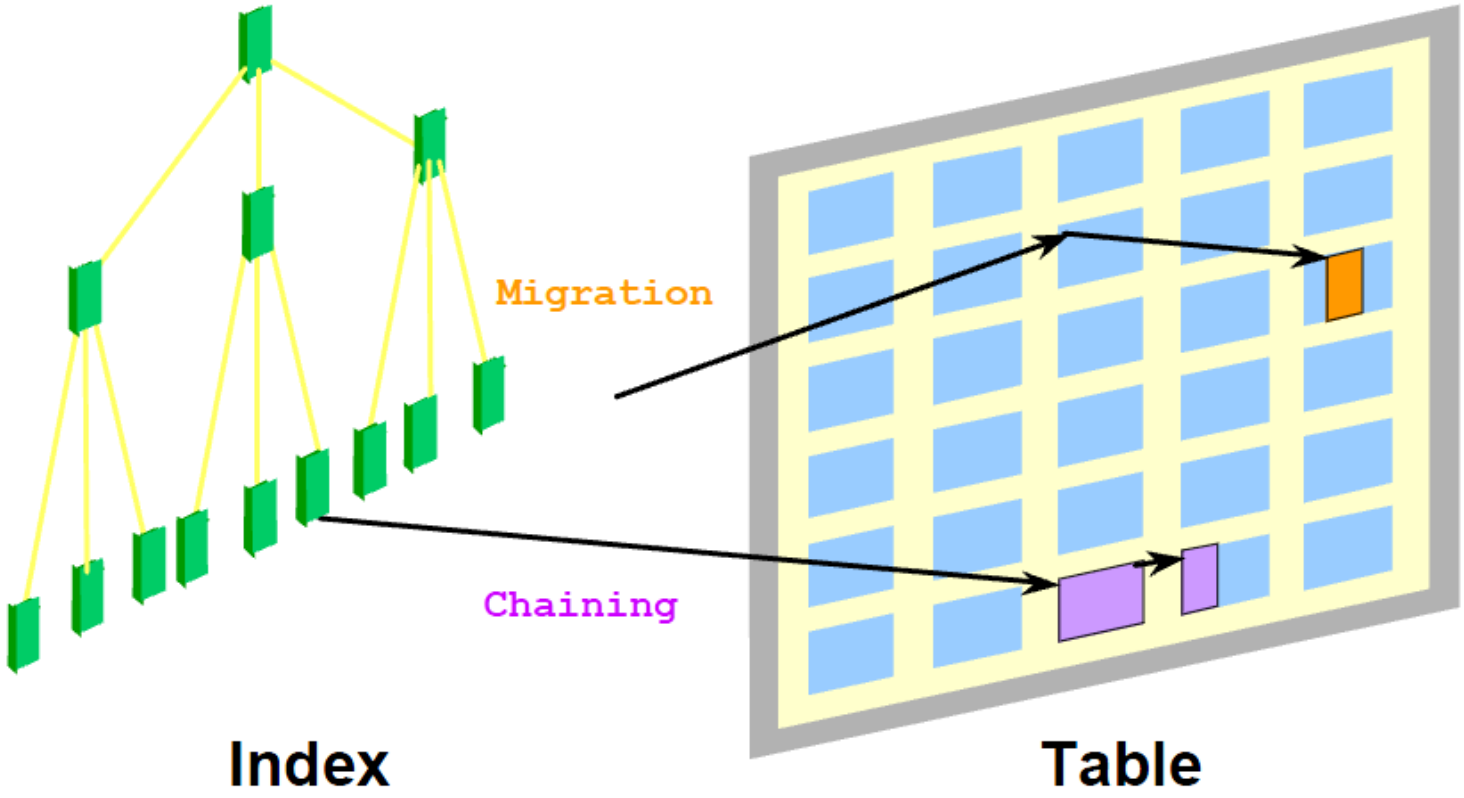



图3-17 偏移量表让我们在块中滑动记录以便为新记录腾出空间

# Oracle的行链和行迁移现象



# 删除与修改记录

- 空余空间管理的问题：滑动重整，首部登记
- 删除标记的选择
- 变长记录的修改会导致比较复杂的问题


图3-19 记录1可被替代，但是删除标记保留；记录2没有删除标记，可以通过跟踪指向它的指针而看到它

# Oracle的内置数据类型

- CHAR, NCHAR, VARCHAR2, NVARCHAR2
- NUMBER
- RAW
- LONG, LONG RAW
- DATE
- TIMESTAMP
- INTERVAL
- CLOB, BLOB, NCLOB, BFILE
- ROWID, UROWID

# CHAR

- In a CHAR column, all the empty bytes are padded with spaces (character 32)
- The length is fixed

```
select department_name DNAME, dump(department_name)
from departments;

DNAME       DUMP(DEPARTMENT_NAME)
----------- ----------------------------------------------
ACCOUNTING Typ=96 Len=10: 65,67,67,79,85,78,84,73,78,71
SALES       Typ=96 Len=10: 83,65,76,69,83,32,32,32,32,32
IT          Typ=96 Len=10: 73,84,32,32,32,32,32,32,32,32
PAYROLL     Typ=96 Len=10: 80,65,89,82,79,76,76,32,32,32
```

# VARCHAR2

- In a VARCHAR2 column, the Oracle server does not pad any empty bytes with spaces
- The length is variable

```
select department_name DNAME,dump(department_name)
from departments;

DNAME      DUMP(DEPARTMENT_NAME)
---------- ----------------------------------------
ACCOUNTING Typ=1 Len=10: 65,67,67,79,85,78,84,73,78,71
SALES      Typ=1 Len= 5: 83,65,76,69,83
IT         Typ=1 Len= 2: 73,84
PAYROLL    Typ=1 Len= 7: 80,65,89,82,79,76,76
```

# NUMBER

- Numeric data is stored internally in a variable length array of one-byte digits
- Numeric data follows this byte format:
  `<[length]>,sign bit/exponent,digit1,digit2,…,digit20`
- Internal code: 2

The exponent byte consists of three parts:

- The sign bit, which is the high order bit (128)
  - If not set (0), the number is negative
  - If set (1), the number is positive or zero
- The offset, which is always 65
- The exponent
  - In the range -65 ... 62
  - Exponent of the number is scientific notation base 100

# Number

## Positive Number Examples

```
SQL> select n,dump(n), substr(dump(n,16),13,6) HEX
     from x;
```

```
         N DUMP(N)                    HEX
------- -------------------- ------
         1 Typ=2 Len=2: 193,2   c1,2
        10 Typ=2 Len=2: 193,11  c1,b
       100 Typ=2 Len=2: 194,2   c2,2
      1200 Typ=2 Len=2: 194,13  c2,d
        .1 Typ=2 Len=2: 192,11  c0,b
       .01 Typ=2 Len=2: 192,2   c0,2
      .001 Typ=2 Len=2: 191,11  bf,b
     .0012 Typ=2 Len=2: 191,13  bf,d
```

# Number

## Positive Number Interpretation

```
dump(123456.789, 16)
--------------------------------
Typ=2 Len=6: c3,d,23,39,4f,5b
```

- Exponent => 0xc3 = 195(dec) - 193 = 2
- Digits

0xd = 13(dec) -1 = 12 > 12 * 100^2 = 120000
0x23 = 35(dec) -1 = 34 > 34 * 100^1 = 3400
0x39 = 57(dec) -1 = 56 > 56 * 100^0 = 56
0x4f = 79(dec) -1 = 78 > 78 * 100^-1 = .78
0x5b = 91(dec) -1 = 90 > 90 * 100^-2 = .009
sum = 123456.789

## Negative Number Examples

```
SQL> select n,dump(n), substr(dump(n,16),13,6) HEX
from x;
```

```
      N  DUMP(N)                        HEX
------  ------------------------  --------
     -1  Typ=2 Len=3: 62,100,102  3e,64,66
    -10  Typ=2 Len=3: 62, 91,102  3e,5b,66
   -100  Typ=2 Len=3: 61,100,102  3d,64,66
  -1200  Typ=2 Len=3: 61, 89,102  3d,59,66
    -.1  Typ=2 Len=3: 63, 91,102  3f,5b,66
   -.01  Typ=2 Len=3: 63,100,102  3f,64,66
  -.001  Typ=2 Len=3: 64, 91,102  40,5b,66
 -.0012  Typ=2 Len=3: 64, 89,102  40,59,66
```

# Number

## Negative Number Interpretation

```
dump(-123456.789, 16)
--------------------------------
Typ=2 Len=7: 3c,59,43,2d,17,b,66
```

Exponent => 0x3c= 62(dec) - 60 = 2

- Digits

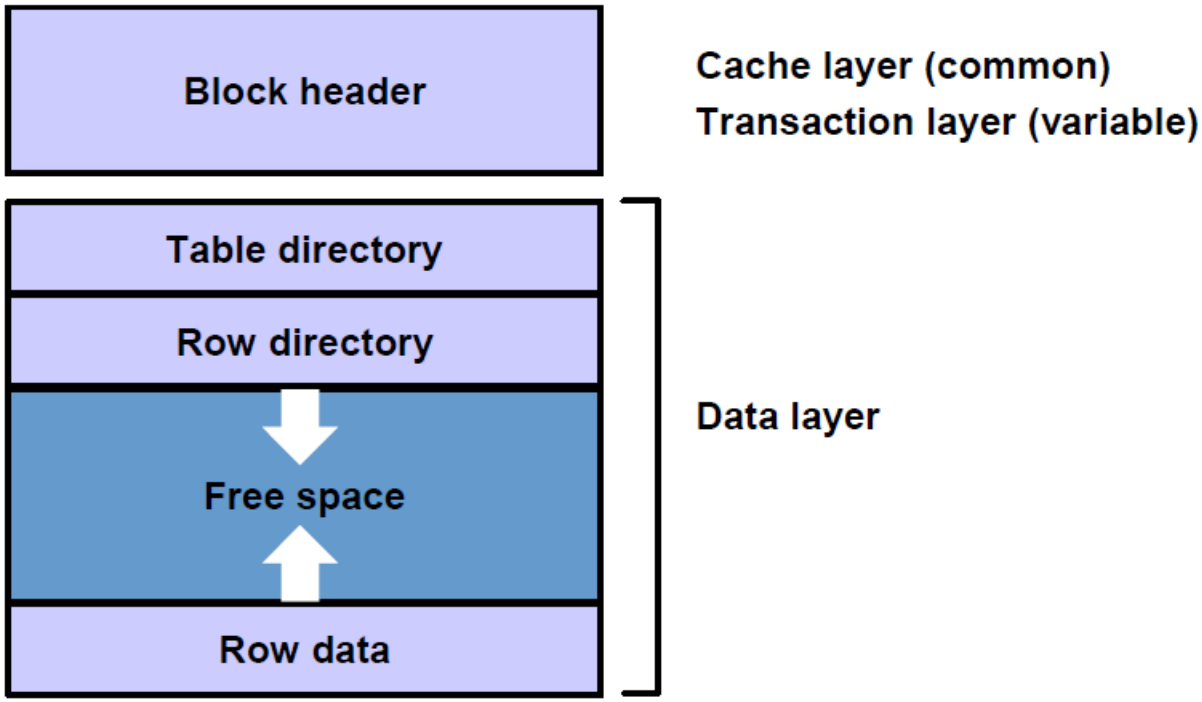
  0x59 = 89(dec): 101 - 89 = 12 > 12 * 100^2 = 120000
  0x43 = 67(dec): 101 - 67 = 34 > 34 * 100^1 = 3400
  0x2d = 45(dec): 101 - 45 = 56 > 56 * 100^0 = 56
  0x17 = 23(dec): 101 - 23 = 78 > 78 * 100^-1 = .78
  0xb = 11(dec): 101 - 11 = 90 > 90 * 100^-2 = .009
  sum = 123456.789 (-)

- Ignore the trailing 0x66 = 102(dec)

# 其它类型

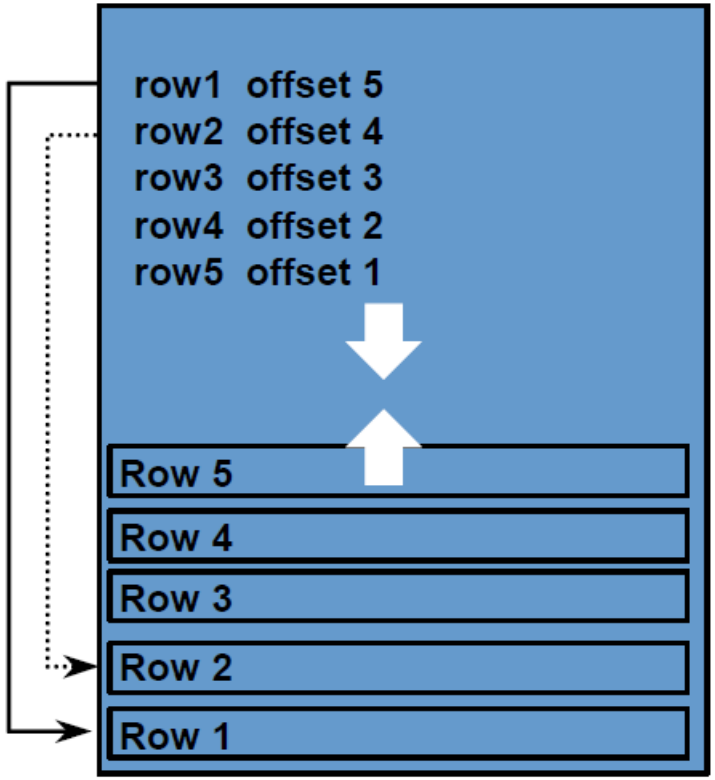
- LONG类型
- RAW类型和LONG RAW
- 日期与时间戳类型

# Oracle的块结构


Block header
Table directory
Row directory
Free space
Row data
Cache layer (common)
Transaction layer (variable)
Data layer

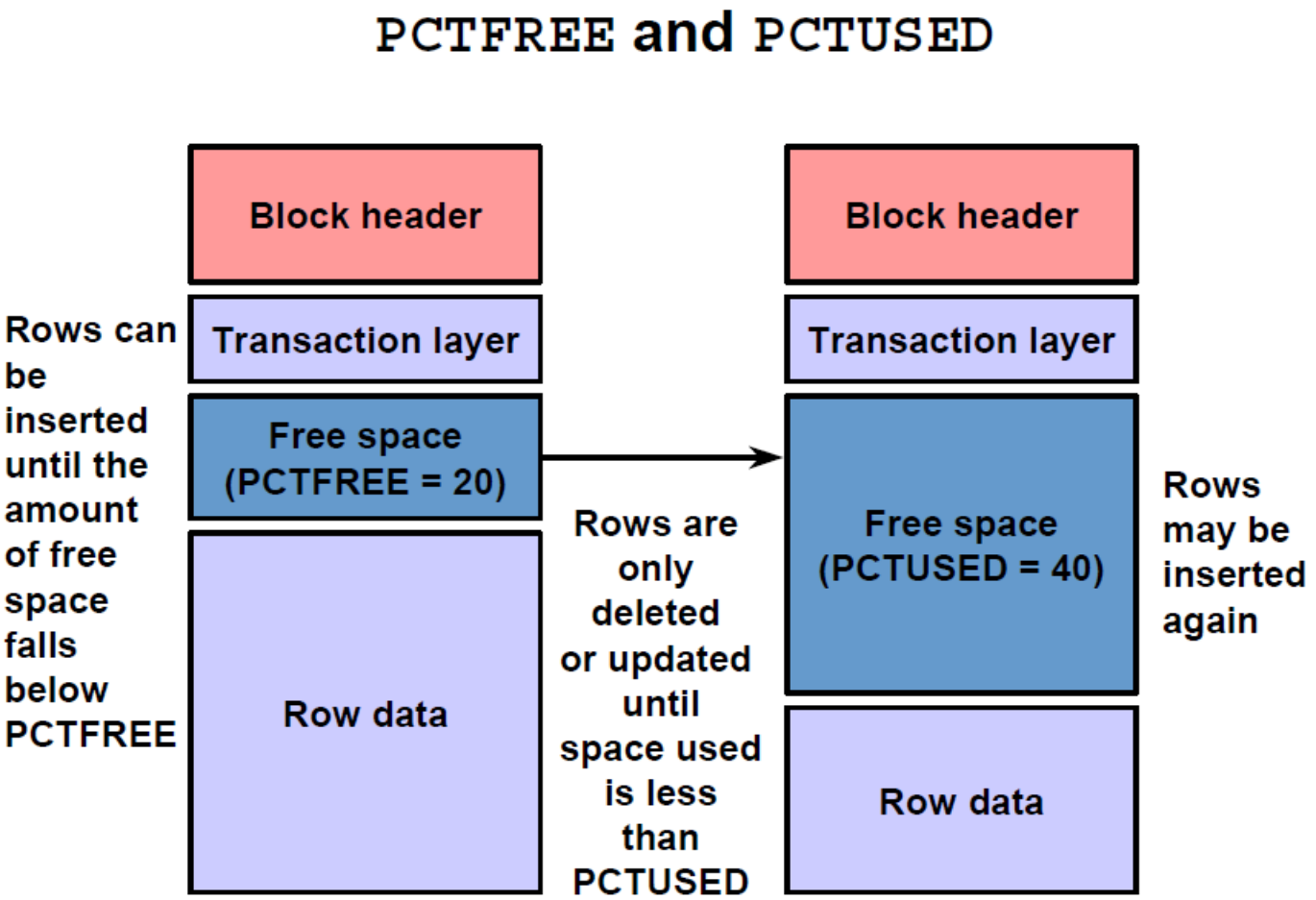
# 数据区

## Block Usage: An Example

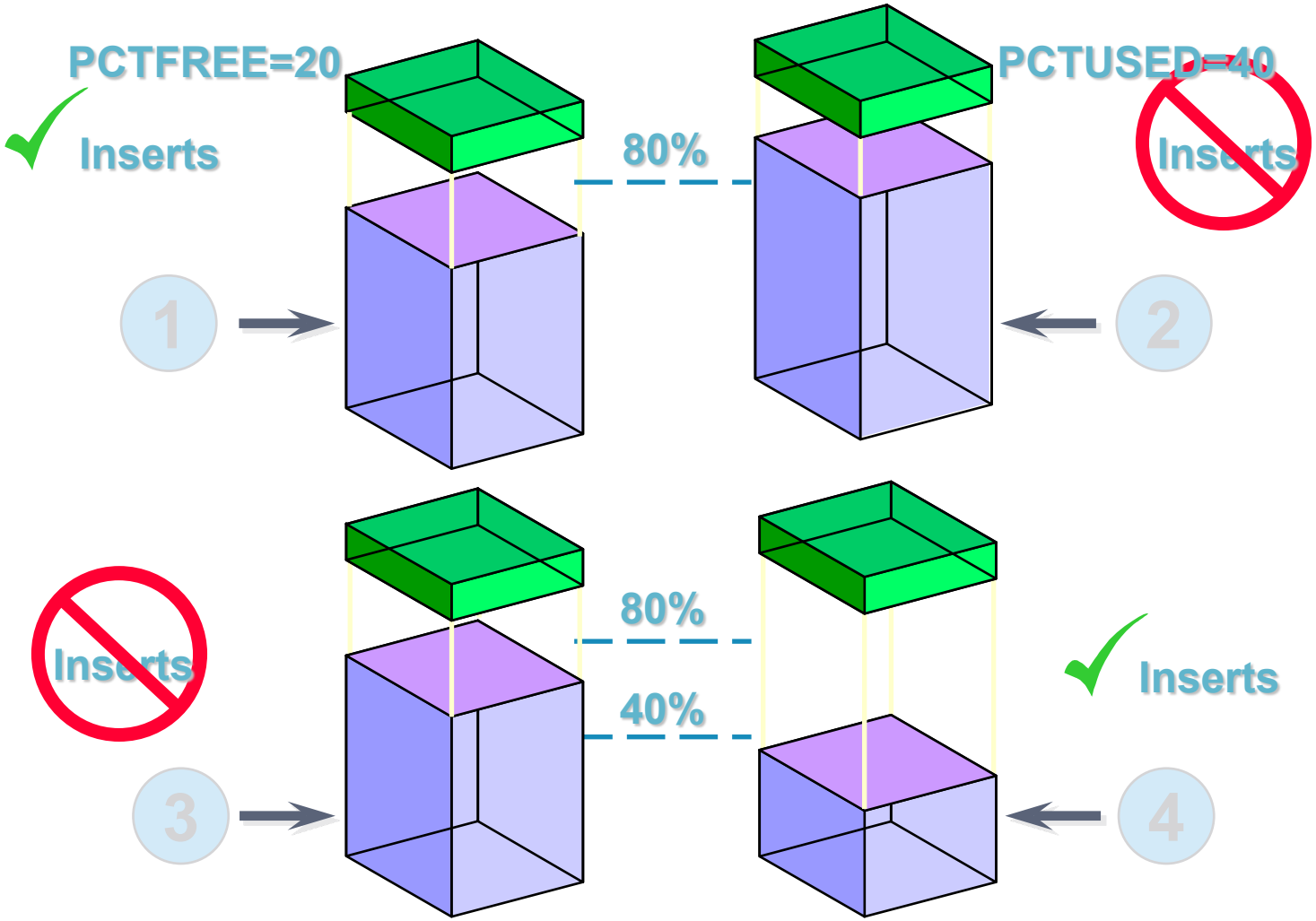



1. Insert from the bottom of the block and move up
2. Try to reuse first free slot in row index
3. Attempt to get space from free space
4. If not enough, compress block (call kdbcps())

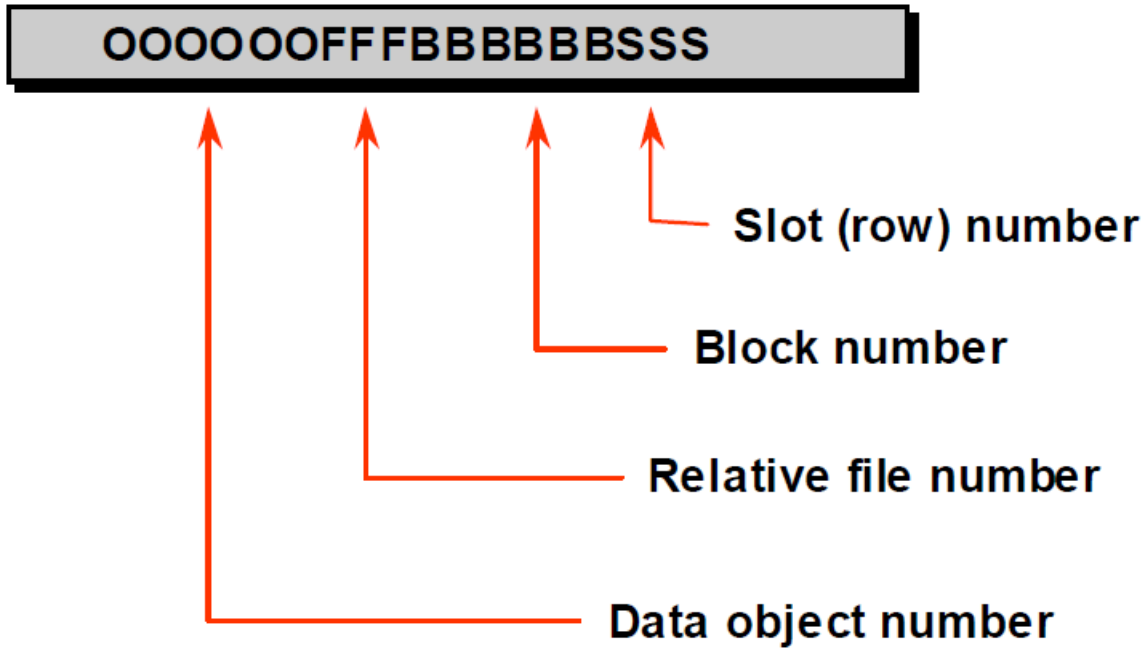
# PCTFREE与PCTUSED存储参数



# Oracle的块策略

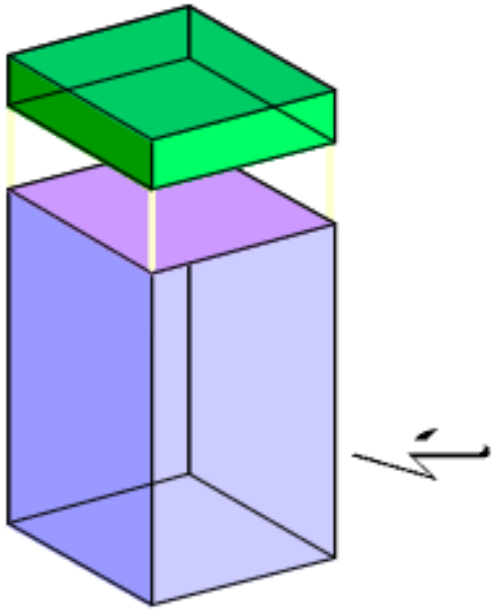

PCTFREE=20
Inserts
80%
PCTUSED=40
Inserts
1
2
Inserts
80%
40%
Inserts
3
4

# ROWID


OOOOOOFFFBBBBBBSSS
Slot (row) number
Block number
Relative file number
Data object number

# 传统数据库的行式存储

- 数据存放在数据文件内
- 数据文件的基本组成单位：块/页
- 块内结构：块头、数据区

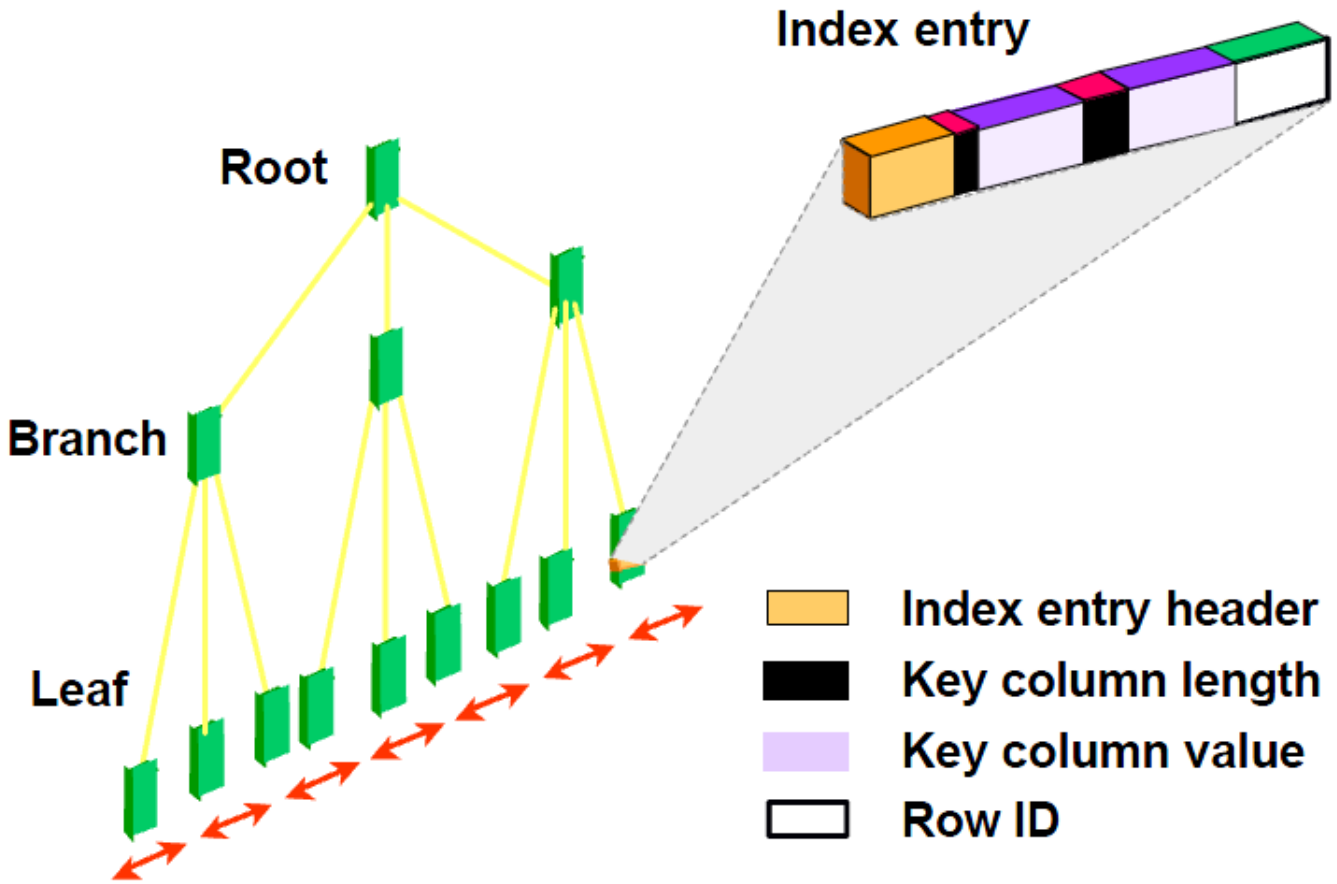


# 行式存储的访问路径

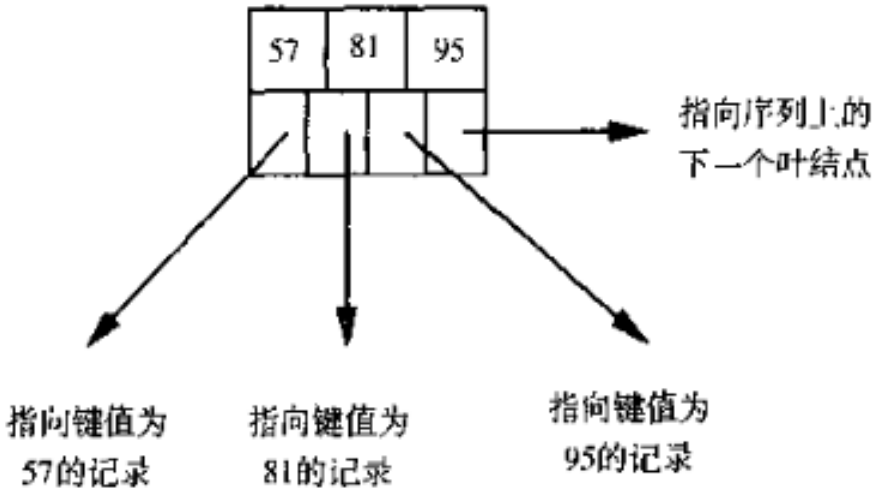
- 全表扫描
- 行标识访问

```
SQL> select rowid,empno,ename,job,sal,deptno from emp;

ROWID                  EMPNO ENAME      JOB              SAL     DEPTNO
------------------ ---------- ---------- --------- ---------- ----------
AAAM9KAAEAAAAJdAAA       7369 SMITH      CLERK            800         20
AAAM9KAAEAAAAJdAAB       7499 ALLEN      SALESMAN        1600         30
AAAM9KAAEAAAAJdAAC       7521 WARD       SALESMAN        1250         30
AAAM9KAAEAAAAJdAAD       7566 JONES      MANAGER         2975         20
AAAM9KAAEAAAAJdAAE       7654 MARTIN     SALESMAN        1250         30
AAAM9KAAEAAAAJdAAF       7698 BLAKE      MANAGER         2850         30
AAAM9KAAEAAAAJdAAG       7782 CLARK      MANAGER         2550         10
AAAM9KAAEAAAAJdAAH       7839 KING       PRESIDENT       8000         10
AAAM9KAAEAAAAJdAAI       7844 TURNER     SALESMAN        1500         30
AAAM9KAAEAAAAJdAAJ       7900 JAMES      CLERK            950         30
AAAM9KAAEAAAAJdAAK       7902 FORD       ANALYST         3000         20
AAAM9KAAEAAAAJdAAL       7934 MILLER     CLERK           1400         10
```

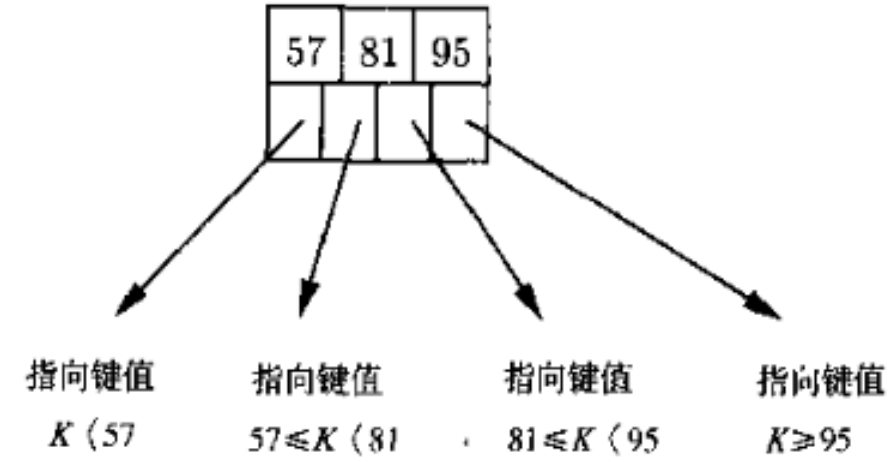
# 行标识访问：B树索引


Index entry
Root
Branch
Leaf
Index entry header
Key column length
Key column value
Row ID

# B树索引原理：结点



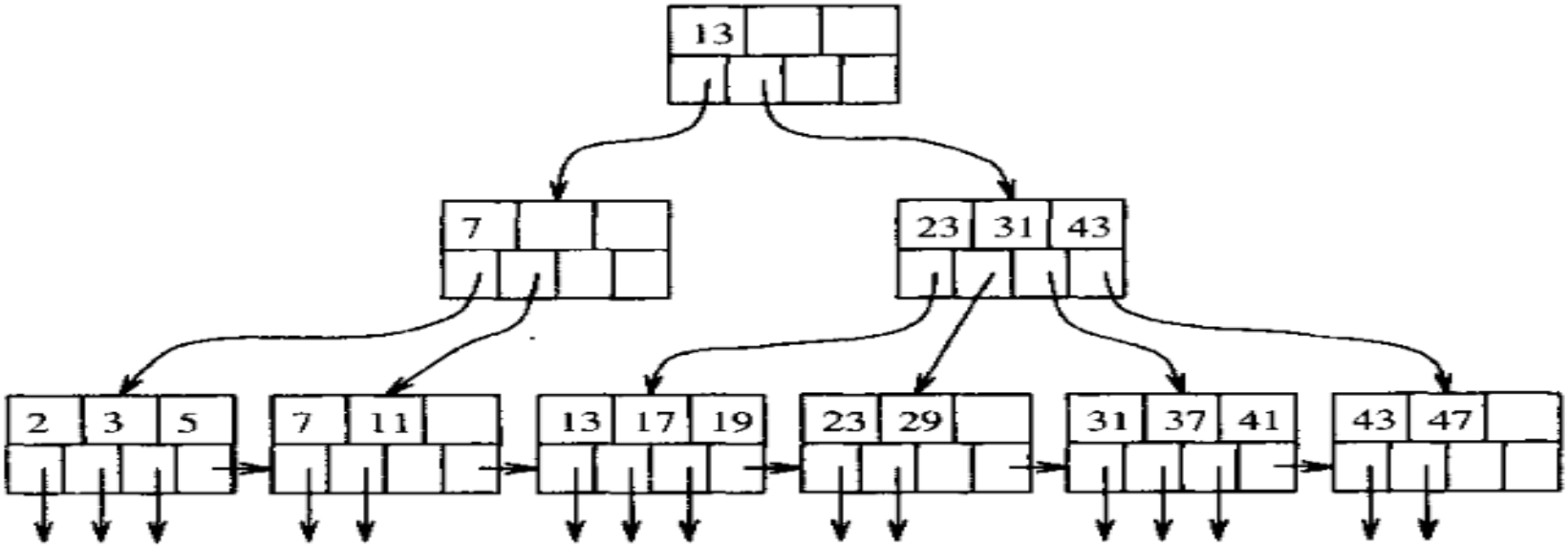

典型的B树叶结点




典型的B树内部结点

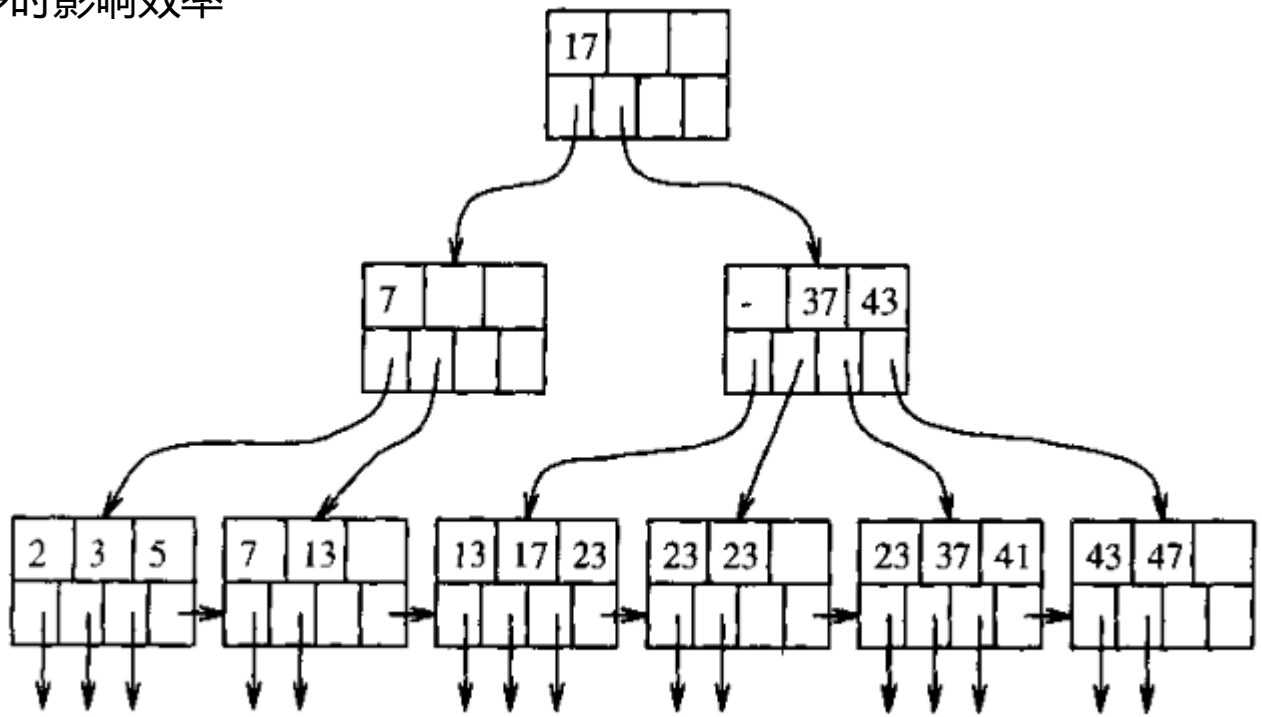
# B树索引原理：树形

- 利用B树进行查询——access path
- B树插入——分裂结点
- B树删除——合并结点



# B树索引的弱点

- 空间代价，创建时间代价，维护代价
- 重复值多时影响效率




一棵带重复键的B树

# 位图索引

- 原理


1 AAAM9KAAEAAAAJdAAM
2 AAAM9KAAEAAAAJdAAA
3 AAAM9KAAEAAAAJdAAB
4 AAAM9KAAEAAAAJdAAC
5 AAAM9KAAEAAAAJdAAD
6 AAAM9KAAEAAAAJdAAE
7 AAAM9KAAEAAAAJdAAF
8 AAAM9KAAEAAAAJdAAG
9 AAAM9KAAEAAAAJdAAH
10 AAAM9KAAEAAAAJdAAI
11 AAAM9KAAEAAAAJdAAJ
12 AAAM9KAAEAAAAJdAAK
13 AAAM9KAAEAAAAJdAAL
行号-行标识表
B树
SMITH
ALLEN
WARD
JONES
位向量
1000000000001000000010001
0000001000000000000000100
0010000010000000100000100
0001000100010000000100000
位图

# 位向量压缩：分段长度编码

- i，j的意义及压缩算法

**例** 如果$i=13$，那么$j=4$。即我们需要4位二进制来表示$i$。因此，$i$的编码开始部分为1110。我们把$i$的二进制数1101加上，这样，13的编码就是11101101。

$i=1$的编码是01，而$i=0$的编码是00。在每一种情况下，$j=1$，因此我们以一个0开始且0后面为表示$i$的一位二进制数。 □

**例** 让我们来对序列11101101001011进行解码。从第一位开始，我们在第四位上找到第一个0，因而$j=4$，下面四位为1101，因而我们确定第一个整数是13。现在我们剩下001011要解码。

由于第一位是0，我们知道下一位表示整数本身，该整数为0。因此已经解码的序列为13和0，我们必须解码剩下的序列1011。

我们在第二个位置上找到第一个0，于是下结论：最末两位表示最后的整数3。我们的整个分段长度序列是这样：13,0,3。从这些数字中，我们能够重新构造实际的位向量：0000000000000110001。 □

# 位向量维护

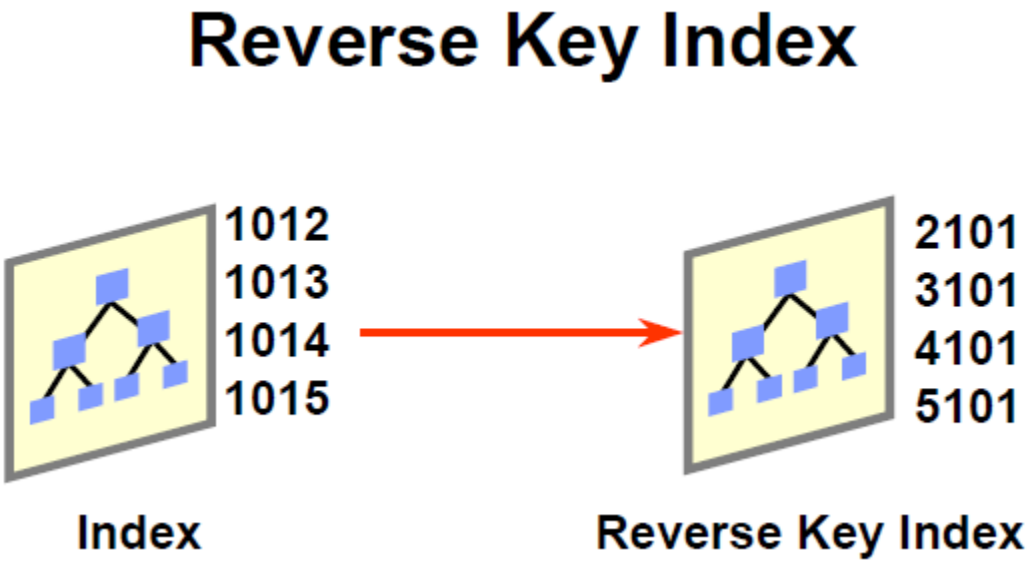
- 删除：位向量相应位置改0，行号-行标识表打上删除标记
- 插入：只需修改跟key值有关的位向量，原因是压缩算法忽略位向量最后连续的0，在行号-行标识里记录新行
- 修改：只需修改有关的2行

# Oracle的索引

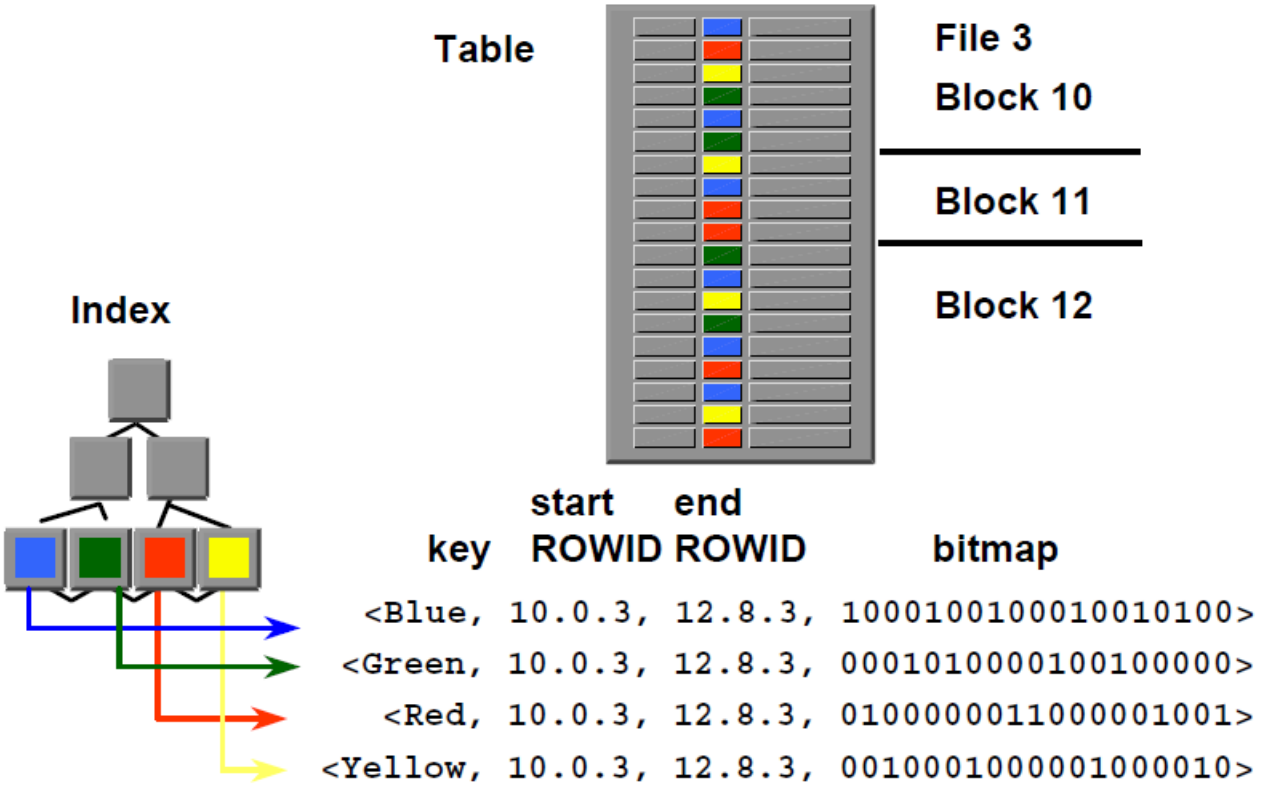
- 支持包括B+树等多种索引
- 创建后自动维护
- 创建时会被加上表级DML共享锁，以及DDL排他锁
- 重建会使用临时表以减少锁表时间

# 反向索引

- 用于key天然形成排序的情形
- 解决局部热点问题

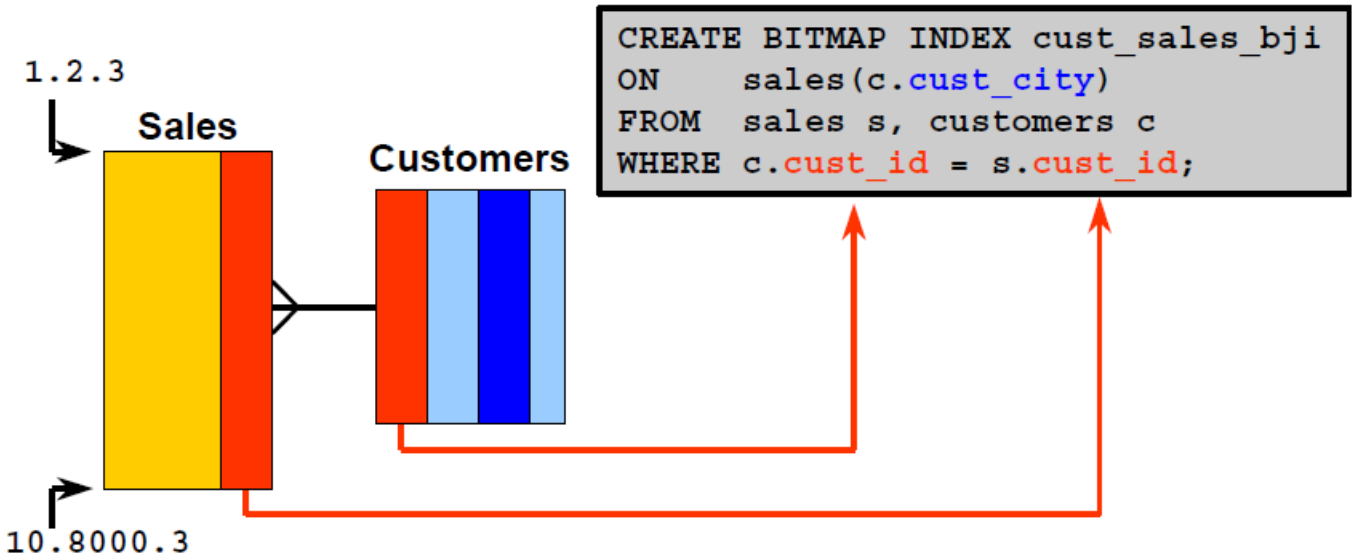


# Oracle的位图索引

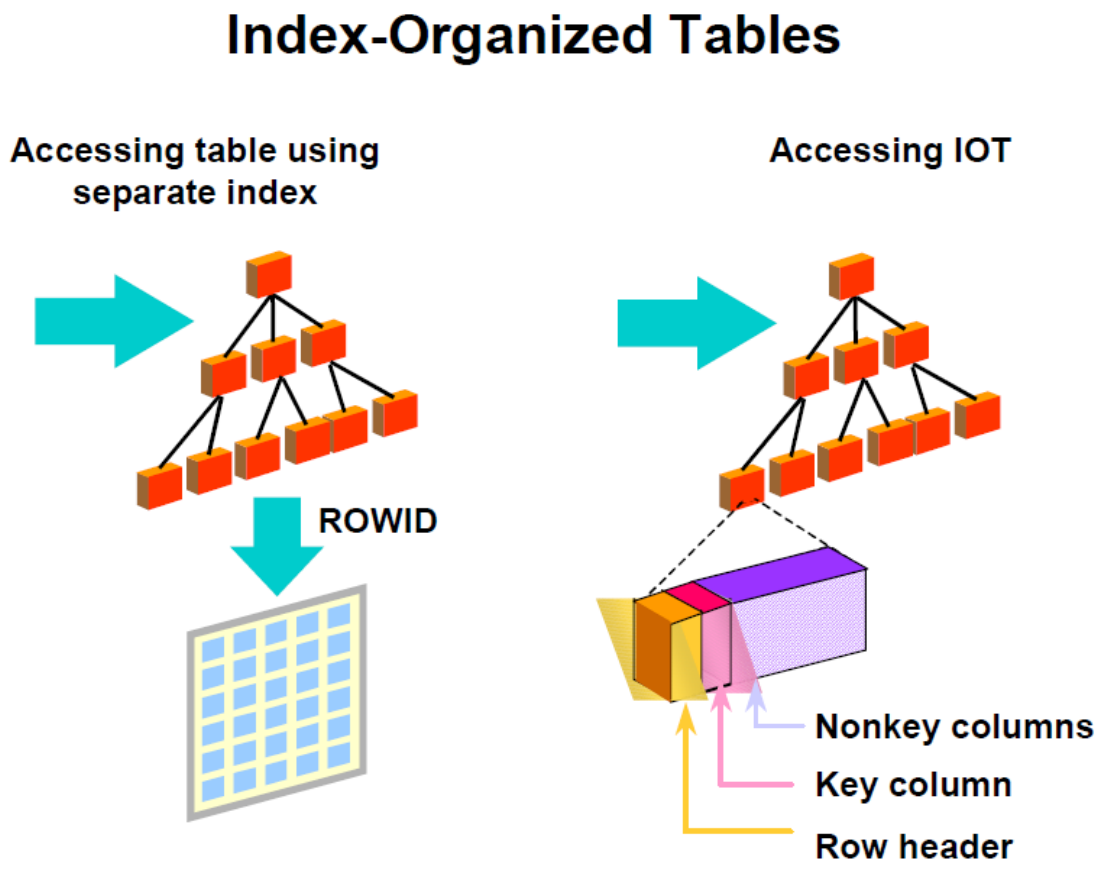


# 位图连接索引

## Bitmap Join Index



# Oracle的索引组织表（IOT）

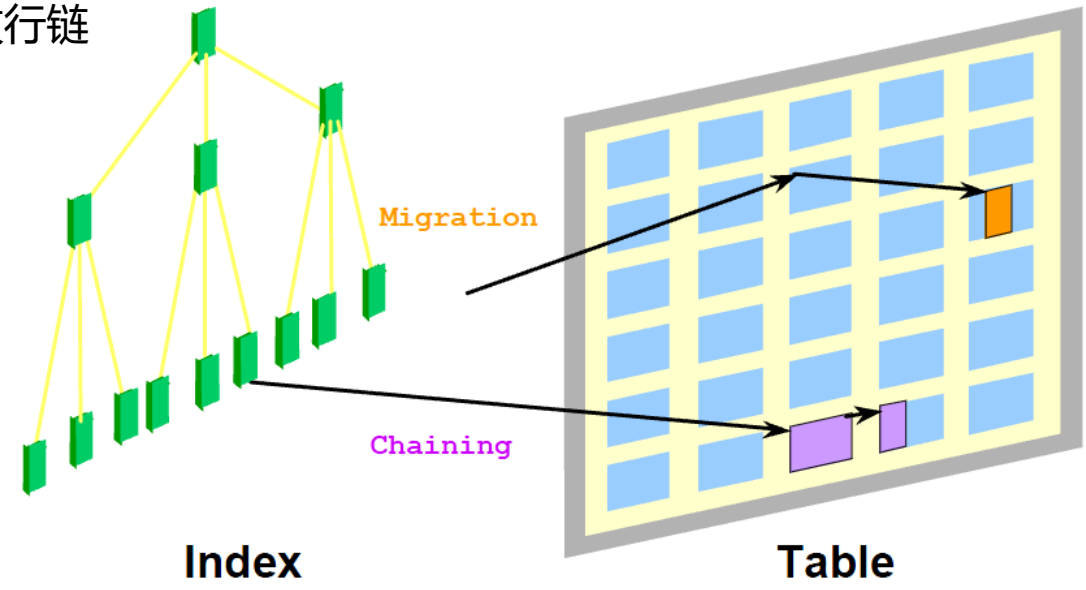


# Oracle的函数索引

- 什么是函数索引？有什么用途
- 打开函数索引功能
- 创建函数索引

# 行式存储的问题

- 读某个列必须读入整行
- 行不等长，修改数据可能导致行迁移
- 行数据较多时可能导致行链



# 数据字典

- 什么是数据字典?
- 数据地图
- 数据字典视图的命名规律
- 其它了解数据字典的方法

# 数据地图

```
SQL> desc dict
 名称                                                                    是否为空? 类型
 ----------------------------------------------------------------------- -------- ----------------------
 TABLE_NAME                                                                       VARCHAR2(30)
 COMMENTS                                                                         VARCHAR2(4000)

SQL> select count(*) from dict;

  COUNT(*)
----------
      1871

SQL> select count(*) from dict where table_name like 'DBA%';

  COUNT(*)
----------
       516

SQL> select count(*) from dict where table_name like 'ALL%';

  COUNT(*)
----------
       272

SQL> select count(*) from dict where table_name like 'USER%'
  2  ;

  COUNT(*)
----------
       285
```

# 数据字典视图的命名规律

- DBA_xxxx
- ALL_xxxx
- USER_xxxx

# 数据字典应用

- 列出用户拥有的表
- 列出用户拥有的表中的列
- 观看用户拥有的特定对象

# 数据字典应用

- 查看主键等约束
- 查看索引

# 数据字典应用

- 观看用户信息
- 观看用户的对象特权
- 观看用户的系统特权
- 观看用户的角色


# Thanks

## FAQ时间