Oracle DUL 程序设计
（节选）

作者　Tomcoding

2017 年 8 月 4 日

前言

如果你看到“前言”两个字就想跳过去，那说明你读这类的文档太多了。照例，在文档的最开始会说明一下项目的背景，让读者能多少了解一下项目的情况，但大多数人在读文档时习惯性的把前言略过去，因为觉得是为凑足字数而写的废话。我们的前言会写短一些，希望你能读一下，当然，略过去也没关系，不会影响你看下面更感兴趣的内容。

在 2005 年的时候，给一个客户做项目，用的是 Oracle 9i 数据库，每天晚上跑批量处理，完了以后要把当天的数据备份出来，当时采用的方案是调用 Oracle 自己的 exp 程序，把整个用户数据全部导出来，然后压缩存储。项目上线后的开始一段时间，数据量比较小，导出非常快，客户的运维人员很快就能做完当日的工作。但是，随着系统运行，数据量也上来了，日终批量处理越来越慢，数据备份也越来越慢，维护人员不得不晚回家，抱怨就开始了。

当时希望找到一种方法能够把数据备份的时间缩短，评估过很多方案，其中之一就是写一个程序从 Oracle 的数据库中直接读取文件中的数据，被认为是最快的途径，但也是难度最大的。首先要搞清楚 Oracle 的数据是怎样存储的，数据类型的表达方式，存储分配的管理等等，后面还要搞清楚分区表、分区之间的存储关联关系，当然，这个方案提出来，根本没讨论上面这些问题就被否决了，因为在当时的技术水平看来，根本不可能做出来。最后，还是购买了一个备份软件来完成这项工作。

如果一个程序员因为技术能力不够而不能完成某项工作，会严重打击自尊和自信。如果遇到暂时无法完成的任务，一定不要放弃。后来在一个偶然的机会，看到有的公司做到了从 Oracle 数据库直接读取文件中的数据，心中燃起了一丝希望，榜样的力量是巨大的，在别人能做到的情况下，你的自信也就增强了。后来也慢慢知道 Oracle 自己有一个叫 DUL 的工具，就是在 Oracle 数据库不能正常启动时，直接从数据文件中获取数据。从那时起，开始对 Oracle 数据库的存储结构进行深入的学习，慢慢掌握了一些基础知识，为自己能够写一个类似的软件打一个基础，也对 Oracle 数据库有了更深入的理解。

遇到的问题是一个人主动学习的推动力，越是困难的问题越能促使自己学习

更多的知识，所以不要放过难题，即使不能立即解决，也要永存在心，当你学到足够的知识时，问题自然就能解决，自身的水平也会随之提高。

从哪儿开始

我们在另一篇文档中已经分析了 Oracle 数据库一些对象的存储结构了，在下面的设计中，这是主要的参考资料。如果要从数据文件中读取数据，你一定会问自己几个问题，从哪儿开始读？读到哪儿结束？对象和存储之间有什么对应关系？

要回答这些问题，就要了解 Oracle 的数据字典，我们先看一下数据字典是个什么样子，都包含哪些数据，然后，我们再仔细考察一下与存储有关的数据字典，就会清楚我们该怎样去做。

数据字典

数据字典是 Oracle 数据库存储元数据的对象，由一系列的基础表和视图组成。我们把与开发这个软件有关的基础表和视图摘录出来，放到下面的列表中，便于后面分析我们感兴趣的内容。我们不需要把这些基础表的关系都搞清楚，只需要把能回答我们上面问题的表关系理清即可。

下面的内容有点多，都是从 Oracle 创建数据库的 SQL 语句中抄录过来的，可以先大致浏览一下，也可以跳过去，在需要时再回过头来查找某项解释。

基础表

tab$

表对象的基表，每个表创建时，都会插入一行，包含一个表的基本信息。

列名	类型	含义
OBJ#	NUMBER	object number
DATAOBJ#	NUMBER	data layer object number
TS#	NUMBER	tablespace number
FILE#	NUMBER	segment header file number

BLOCK#	NUMBER	segment header block number
BOBJ#	NUMBER	base object number (cluster / iot)
TAB#	NUMBER	table number in cluster, NULL if not clustered
COLS	NUMBER	number of columns
CLUCOLS	NUMBER	number of clustered columns, NULL if not clustered
PCTFREE$	NUMBER	minimum free space percentage in a block
PCTUSED$	NUMBER	minimum used space percentage in a block
INITRANS	NUMBER	initial number of transaction
MAXTRANS	NUMBER	maximum number of transaction
FLAGS	NUMBER	0x00000000 = unmodified since last backup 0x00000001 = modified since then 0x00000002 = DML locks restricted to <= SX 0x00000004 = DML locks <= SX not acquired 0x00000008 = CACHE 0x00000010 = table has been analyzed 0x00000020 = table has no logging 0x00000040 = 7.3 -> 8.0 data object migration required 0x00000080 = current summary dependency 0x00000100 = user-specified stats 0x00000200 = global stats 0x00000800 = table has security policy 0x00020000 = Move Partitioned Rows 0x00400000 = table has sub tables 0x00800000 = row dependencies enabled 0x10000000 = this IOT has a physical rowid mapping table 0x20000000 = mapping table of an IOT (with physical rowid)
AUDIT$	VARCHAR2(38)	auditing options
ROWCNT	NUMBER	number of rows
BLKCNT	NUMBER	number of blocks
EMPCNT	NUMBER	number of empty blocks
AVGSPC	NUMBER	average available free space/iot overflow stats
CHNCNT	NUMBER	number of chained rows
AVGRLN	NUMBER	average row length
AVGSPC_FLB	NUMBER	average available free space of blocks on free list
FLBCNT	NUMBER	free list block count
ANALYZETIME	DATE	timestamp when last analyzed
SAMPLESIZE	NUMBER	number of rows sampled by Analyze
DEGREE	NUMBER	number of parallel query slaves per instance
INSTANCES	NUMBER	number of OPS instances for parallel query
INTCOLS	NUMBER	number of internal columns
KERNELCOLS	NUMBER	number of REAL (kernel) columns

PROPERTY	NUMBER	0x00000001 = typed table 0x00000002 = has ADT columns 0x00000004 = has nested-TABLE columns 0x00000008 = has REF columns 0x00000010 = has array columns 0x00000020 = partitioned table 0x00000040 = index-only table (IOT) 0x00000080 = IOT w/ row OVerflow 0x00000100 = IOT w/ row CLustering 0x00000200 = IOT OVeRflow segment 0x00000400 = clustered table 0x00000800 = has internal LOB columns 0x00001000 = has primary key-based OID$ column 0x00002000 = nested table 0x00004000 = View is Read Only 0x00008000 = has FILE columns 0x00010000 = obj view's OID is system-gen 0x00020000 = used as AQ table 0x00040000 = has user-defined lob columns 0x00080000 = table contains unused columns 0x00100000 = has an on-commit materialized view 0x00200000 = has system-generated column names 0x00400000 = global temporary table 0x00800000 = session-specific temporary table 0x08000000 = table is a sub table 0x20000000 = pdml itl invariant 0x80000000 = table is external
TRIGFLAG	NUMBER	0x00000001 = deferred RPC Queue 0x00000002 = snapshot log 0x00000004 = updatable snapshot log 0x00000008 = context trigger 0x00000010 = synchronous change table 0x00000020 = Streams trigger 0x00000040 = Content Size Trigger 0x00000080 = audit vault trigger 0x00000100 = Streams Auxiliary Logging trigger 0x00010000 = server-held key encrypted columns exist 0x00020000 = user-held key encrypted columns exist 0x00200000 = table is read only 0x00400000 = lobs use shared segment 0x00800000 = queue table 0x10000000 = streams unsupported table enabled at some point in past 0x80000000 = Versioning enabled on this table

SPARE1	NUMBER	used to store hakan_kqldtvc
SPARE2	NUMBER	committed partition # used by drop column
SPARE3	NUMBER	summary sequence number
SPARE4	VARCHAR2(1000)	committed RID used by drop column
SPARE5	VARCHAR2(1000)	summary related information on table
SPARE6	DATE	flashback timestamp

clu$

聚簇（cluster）的基表，没创建一个 cluster 会插入一行，包含 cluster 的基本信息。

列名	类型	含义
OBJ#	NUMBER	object number
DATAOBJ#	NUMBER	data layer object number
TS#	NUMBER	tablespace number
FILE#	NUMBER	segment header file number
BLOCK#	NUMBER	segment header block number
COLS	NUMBER	number of columns
PCTFREE$	NUMBER	minimum free space percentage in a block
PCTUSED$	NUMBER	minimum used space percentage in a block
INITRANS	NUMBER	initial number of transaction
MAXTRANS	NUMBER	maximum number of transaction
SIZE$	NUMBER	if b-tree, estimated number of bytes for each cluster key and rows
HASHFUNC	VARCHAR2(30)	if hashed, function identifier
HASHKEYS	NUMBER	hash key count
FUNC	NUMBER	function: 0 (key is function), 1 (system default)
EXTIND	NUMBER	extent index value of fixed hash area
FLAGS	NUMBER	0x00000008 = CACHE 0x00010000 = Single Table Cluster 0x00800000 = DEPENDENCIES
DEGREE	NUMBER	number of parallel query slaves per instance
INSTANCES	NUMBER	number of OPS instances for parallel query
AVGCHN	NUMBER	average chain length - previously spare4
SPARE1	NUMBER	used for trigger non-trigger flags 0x00010001 = replication 0x00010002 = snapshot log 0x00010004 = snapshot 0x00010008 = context internal trigger 0x00000010 = synchronous change table 0x00010000 = One or more columns are encrypted 0x00020000 = All columns are encrypted

		0x00040000 = needs to do logging 0x00080000 = MV Dataless 0x00100000 = IOT transient table for PMO 0x00200000 = MPR 0x00400000 = QUEue organized table
SPARE2	NUMBER	
SPARE3	NUMBER	
SPARE4	NUMBER	
SPARE5	VARCHAR2(1000)	
SPARE6	VARCHAR2(1000)	
SPARE7	DATE	

seg$

段对象（segment）的基表，每个段一行。

列名	类型	含义
FILE#	NUMBER	segment header file number
BLOCK#	NUMBER	segment header block number
TYPE#	NUMBER	segment type (see KTS.H) 1 = UNDO 2 = SAVE UNDO 3 = TEMPORARY 4 = CACHE 5 = DATA 6 = INDEX 7 = SORT 8 = LOB 9 = Space Header 10 = System Managed Undo
TS#	NUMBER	tablespace containing this segment
BLOCKS	NUMBER	blocks allocated to segment so far zero for bitmapped tablespaces
EXTENTS	NUMBER	extents allocated to segment so far zero for bitmapped tablespaces
INIEXTS	NUMBER	initial extent size
MINEXTS	NUMBER	minimum number of extents
MAXEXTS	NUMBER	maximum number of extents
EXTSIZE	NUMBER	current next extent size zero for bitmapped tablespaces
EXTPCT	NUMBER	percent size increase
USER#	NUMBER	user who owns this segment
LISTS	NUMBER	freelists for this segment

GROUPS	NUMBER	freelist groups for this segment
BITMAPRANGES	NUMBER	ranges per bit map entry
CACHEHINT	NUMBER	hints for caching
SCANHINT	NUMBER	hints for scanning
HWMINCR	NUMBER	Amount by which HWM is moved
SPARE1	NUMBER	0x00000001 - bitmapped tablespace: KTSSEGM_FLAG_BITMAPPED_TS 0x00000002 - undo segment: KTSSEGM_FLAG_UNDOSEG 0x00000004 - saveundo segment: KTSSEGM_FLAG_SVUNDOSEG 0x00000008 - segment marked corrupt: KTSSEGM_FLAG_CORRUPT 0x00000010 - KTSSEGM_FLAG_BEING_MIGRATED 0x00000020 - KTSSEGM_FLAG_BMAPHDR_SEG 0x00000040 - KTSSEGM_FLAG_BLKSFILLED 0x00000080 - KTSSEGM_FLAG_NOQUOTA 0x00000100 - KTSSEGM_FLAG_PAGETABLE 0x00000200 - KTSSEGM_FLAG_TEMPOBJ 0x00000400 - KTSSEGM_FLAG_OBJDEFINED 0x00000800 - KTSSEGM_FLAG_COMPRESSED 0x00001000 - KTSSEGM_FLAG_HASCPRSSED 0x00002000 - KTSSEGM_FLAG_ROWMOVEMNT 0x00004000 - KTSSEGM_FLAG_HASMOVEMNT 0x00010000 - segment flushed from cache: KTSSEGM_FLAG_RECYCLEBIN 0x00040000 - 11gR1 HSC Compressed Segment: KTSSEGM_FLAG_NEW_HSC 0x00080000 - 11gR1 LZO Compressed Segment: KTSSEGM_FLAG_LZO 0x00100000 - 11gR1 ZLIB Compressed Segmnt: KTSSEGM_FLAG_ZLIB 0x00200000 - Next generation LOB segment: KTSSEGM_FLAG_NGLOB 0x00400000 - Segment has MAXSIZE set: KTSSEGM_FLAG_MAXSZSET 0x00800000 - Encrypted segment: KTSSEGM_FLAG_ENC 0x01000000 - OLTP Compressed: KTSSEGM_FLAG_OLTP 0x02000000 - Columnar Compressed low: KTSSEGM_FLAG_ARCH1 0x04000000 - Columnar Compressed high: KTSSEGM_FLAG_ARCH2

		0x08000000 - Columnar Compressed archive: KTSSEGM_FLAG_ARCH3 0x10000000 - Read mostly segment: KTSSEGM_FLAG_READMOSTLY
SPARE2	NUMBER	

file$

数据文件（file）的基表，每个数据文件一行。

列名	类型	含义
FILE#	NUMBER	file identifier number
STATUS$	NUMBER	status (see KTS.H): 1 = INVALID, 2 = AVAILABLE
BLOCKS	NUMBER	size of file in blocks zero for bitmapped tablespaces
TS#	NUMBER	tablespace that owns file
RELFILE#	NUMBER	relative file number
MAXEXTEND	NUMBER	maximum file size
INC	NUMBER	increment amount
CRSCNWRP	NUMBER	creation SCN wrap
CRSCNBAS	NUMBER	creation SCN base
OWNERINSTANCE	VARCHAR2(30)	Owner instance name
SPARE1	NUMBER	tablespace-relative DBA of space file header NULL for dictionary-mapped tablespaces
SPARE2	NUMBER	
SPARE3	VARCHAR2(1000)	
SPARE4	DATE	

obj$

对象（object）的基表，包含了所有 Oracle 对象的信息。

列名	类型	含义
OBJ#	NUMBER	object number
DATAOBJ#	NUMBER	data layer object number
OWNER#	NUMBER	owner user number
NAME	VARCHAR2(30)	object name
NAMESPACE	NUMBER	namespace of object (see KQD.H): 1 = TABLE/PROCEDURE/TYPE 2 = BODY 3 = TRIGGER 4 = INDEX

		5 = CLUSTER 8 = LOB 9 = DIRECTORY 10 = QUEUE 11 = REPLICATION OBJECT GROUP 12 = REPLICATION PROPAGATOR 13 = JAVA SOURCE 14 = JAVA RESOURCE 58 = (Data Mining) MODEL
SUBNAME	VARCHAR2(30)	subordinate to the name
TYPE#	NUMBER	object type (see KQD.H): 1 = INDEX 2 = TABLE 3 = CLUSTER 4 = VIEW 5 = SYNONYM 6 = SEQUENCE 7 = PROCEDURE 8 = FUNCTION 9 = PACKAGE 10 = NON-EXISTENT 11 = PACKAGE BODY 12 = TRIGGER 13 = TYPE 14 = TYPE BODY 19 = TABLE PARTITION 20 = INDEX PARTITION 21 = LOB 22 = LIBRARY 23 = DIRECTORY 24 = QUEUE 25 = IOT 26 = REPLICATION OBJECT GROUP 27 = REPLICATION PROPAGATOR 28 = JAVA SOURCE 29 = JAVA CLASS 30 = JAVA RESOURCE 31 = JAVA JAR 32 = INDEXTYPE 33 = OPERATOR 34 = TABLE SUBPARTITION 35 = INDEX SUBPARTITION 82 = (Data Mining) MODEL 92 = OLAP CUBE DIMENSION

		93 = OLAP CUBE 94 = OLAP MEASURE FOLDER 95 = OLAP CUBE BUILD PROCESS
CTIME	DATE	object creation time
MTIME	DATE	DDL modification time
STIME	DATE	specification timestamp (version)
STATUS	NUMBER	status of object (see KQD.H): 1 = VALID/AUTHORIZED WITHOUT ERRORS 2 = VALID/AUTHORIZED WITH AUTHORIZATION ERRORS 3 = VALID/AUTHORIZED WITH COMPILATION ERRORS 4 = VALID/UNAUTHORIZED 5 = INVALID/UNAUTHORIZED
REMOTEOWNER	VARCHAR2(30)	remote owner name (remote object)
LINKNAME	VARCHAR2(128)	link name (remote object)
FLAGS	NUMBER	0x0001 = extent map checking required 0x0002 = temporary object 0x0004 = system generated object 0x0008 = unbound (invoker's rights) 0x0010 = secondary object 0x0020 = in-memory temp table 0x0080 = dropped table (RecycleBin) 0x0100 = synonym VPD policies 0x0200 = synonym VPD groups 0x0400 = synonym VPD context 0x4000 = nested table partition
OID$	RAW(16)	OID for typed table, typed view, and type
SPARE1	NUMBER	sql version flag: see kpul.h
SPARE2	NUMBER	object version number
SPARE3	NUMBER	base user#
SPARE4	VARCHAR2(1000)	
SPARE5	VARCHAR2(1000)	
SPARE6	DATE	

col$

列（column）的基表，包含所有表的列信息，也包含聚簇（cluster）列的信息。

列名	类型	含义
OBJ#	NUMBER	object number of base object
COL#	NUMBER	column number as created
SEGCOL#	NUMBER	column number in segment
SEGCOLLENGTH	NUMBER	length of the segment column
OFFSET	NUMBER	offset of column

NAME	VARCHAR2(30)	name of column
TYPE#	NUMBER	data type of column for ADT column, type# = DTYADT
LENGTH	NUMBER	length of column in bytes
FIXEDSTORAGE	NUMBER	flags: 0x01 = fixed, 0x02 = read-only
PRECISION#	NUMBER	precision
SCALE	NUMBER	scale
NULL$	NUMBER	0 = NULLs permitted, >0 = no NULLs permitted
DEFLENGTH	NUMBER	default value expression text length
DEFAULT$	LONG	default value expression text
INTCOL#	NUMBER	internal column number
PROPERTY	NUMBER	column properties (bit flags): 0x00000001 = ADT attribute column 0x00000002 = OID column 0x00000004 = nested table column 0x00000008 = virtual column 0x00000010 = nested table's SETID$ column 0x00000020 = hidden column 0x00000040 = primary-key based OID column 0x00000080 = column is stored in a lob 0x00000100 = system-generated column 0x00000200 = rowinfo column of typed table/view 0x00000400 = nested table columns setid 0x00000800 = column not insertable 0x00001000 = column not updatable 0x00002000 = column not deletable 0x00004000 = dropped column 0x00008000 = unused column - data still in row 0x00010000 = virtual column 0x00020000 = place DESCEND operator on top 0x00040000 = virtual column is NLS dependent 0x00080000 = ref column (present as oid col) 0x00100000 = hidden snapshot base table column 0x00200000 = attribute column of a user-defined ref 0x00400000 = export hidden column,RLS on hidden col 0x00800000 = string column measured in characters 0x01000000 = virtual column expression specified 0x02000000 = typeid column 0x04000000 = Column is encrypted 0x20000000 = Column is encrypted without salt
CHARSETID	NUMBER	NLS character set id
CHARSETFORM	NUMBER	1 = implicit: for CHAR, VARCHAR2, CLOB w/o a specified set 2 = nchar: for NCHAR, NCHAR VARYING, NCLOB 3 = explicit: for CHAR, etc. with "CHARACTER SET ..." clause

		4 = flexible: for PL/SQL "flexible" parameters
SPARE1	NUMBER	fractional seconds precision
SPARE2	NUMBER	interval leading field precision
SPARE3	NUMBER	maximum number of characters in string
SPARE4	VARCHAR2(1000)	NLS settings for this expression
SPARE5	VARCHAR2(1000)	
SPARE6	DATE	

……部分信息省略 ……

视图

DBA_DATA_FILES

数据文件视图。

```
create or replace view DBA_DATA_FILES
    (FILE_NAME, FILE_ID, TABLESPACE_NAME,
     BYTES, BLOCKS, STATUS, RELATIVE_FNO, AUTOEXTENSIBLE,
     MAXBYTES, MAXBLOCKS, INCREMENT_BY, USER_BYTES, USER_BLOCKS, ONLINE_STATUS)
as
select v.name, f.file#, ts.name,
       ts.blocksize * f.blocks, f.blocks,
       decode(f.status$, 1, 'INVALID', 2, 'AVAILABLE', 'UNDEFINED'),
       f.relfile#, decode(f.inc, 0, 'NO', 'YES'),
       ts.blocksize * f.maxextend, f.maxextend, f.inc,
       ts.blocksize * (f.blocks - 1), f.blocks - 1,
       decode(fe.fetsn, 0, decode(bitand(fe.festa, 2), 0, 'SYSOFF', 'SYSTEM'),
         decode(bitand(fe.festa, 18), 0, 'OFFLINE', 2, 'ONLINE', 'RECOVER'))
from sys.file$ f, sys.ts$ ts, sys.v$dbfile v, x$kccfe fe
where v.file# = f.file#
  and f.spare1 is NULL
  and f.ts# = ts.ts#
  and fe.fenum = f.file#
union all
select
       v.name,f.file#, ts.name,
       decode(hc.ktfbhccval, 0, ts.blocksize * hc.ktfbhcsz, NULL),
       decode(hc.ktfbhccval, 0, hc.ktfbhcsz, NULL),
       decode(f.status$, 1, 'INVALID', 2, 'AVAILABLE', 'UNDEFINED'),
       f.relfile#,
       decode(hc.ktfbhccval, 0, decode(hc.ktfbhcinc, 0, 'NO', 'YES'), NULL),
       decode(hc.ktfbhccval, 0, ts.blocksize * hc.ktfbhcmaxsz, NULL),
```

```
       decode(hc.ktfbhccval, 0, hc.ktfbhcmaxsz, NULL),
       decode(hc.ktfbhccval, 0, hc.ktfbhcinc, NULL),
       decode(hc.ktfbhccval, 0, hc.ktfbhcusz * ts.blocksize, NULL),
       decode(hc.ktfbhccval, 0, hc.ktfbhcusz, NULL),
       decode(fe.fetsn, 0, decode(bitand(fe.festa, 2), 0, 'SYSOFF', 'SYSTEM'),
         decode(bitand(fe.festa, 18), 0, 'OFFLINE', 2, 'ONLINE', 'RECOVER'))
from sys.v$dbfile v, sys.file$ f, sys.x$ktfbhc hc, sys.ts$ ts, x$kccfe fe
where v.file# = f.file#
  and f.spare1 is NOT NULL
  and v.file# = hc.ktfbhcafno
  and hc.ktfbhctsn = ts.ts#
  and fe.fenum = f.file#
```

DBA_EXTENTS

区（extent）视图。

```
create or replace view DBA_EXTENTS
    (OWNER, SEGMENT_NAME, PARTITION_NAME, SEGMENT_TYPE, TABLESPACE_NAME,
     EXTENT_ID, FILE_ID, BLOCK_ID,
     BYTES, BLOCKS, RELATIVE_FNO)
as
select ds.owner, ds.segment_name, ds.partition_name, ds.segment_type,
       ds.tablespace_name,
       e.ext#, f.file#, e.block#, e.length * ds.blocksize, e.length, e.file#
from sys.uet$ e, sys.sys_dba_segs ds, sys.file$ f
where e.segfile# = ds.relative_fno
  and e.segblock# = ds.header_block
  and e.ts# = ds.tablespace_id
  and e.ts# = f.ts#
  and e.file# = f.relfile#
  and bitand(NVL(ds.segment_flags,0), 1) = 0
  and bitand(NVL(ds.segment_flags,0), 65536) = 0
union all
select
       ds.owner, ds.segment_name, ds.partition_name, ds.segment_type,
       ds.tablespace_name,
       e.ktfbueextno, f.file#, e.ktfbuebno,
       e.ktfbueblks * ds.blocksize, e.ktfbueblks, e.ktfbuefno
from sys.sys_dba_segs ds, sys.x$ktfbue e, sys.file$ f
where e.ktfbuesegfno = ds.relative_fno
  and e.ktfbuesegbno = ds.header_block
```

```
  and e.ktfbuesegtsn = ds.tablespace_id
  and ds.tablespace_id = f.ts#
  and e.ktfbuefno = f.relfile#
  and bitand(NVL(ds.segment_flags, 0), 1) = 1
  and bitand(NVL(ds.segment_flags,0), 65536) = 0
```

DBA_OBJECTS

对象视图。

```
create or replace view DBA_OBJECTS
    (OWNER, OBJECT_NAME, SUBOBJECT_NAME, OBJECT_ID, DATA_OBJECT_ID,
     OBJECT_TYPE, CREATED, LAST_DDL_TIME, TIMESTAMP, STATUS,
     TEMPORARY, GENERATED, SECONDARY, NAMESPACE, EDITION_NAME)
as
select u.name, o.name, o.subname, o.obj#, o.dataobj#,
       decode(o.type#, 0, 'NEXT OBJECT', 1, 'INDEX', 2, 'TABLE', 3, 'CLUSTER',
                       4, 'VIEW', 5, 'SYNONYM', 6, 'SEQUENCE',
                       7, 'PROCEDURE', 8, 'FUNCTION', 9, 'PACKAGE',
                       11, 'PACKAGE BODY', 12, 'TRIGGER',
                       13, 'TYPE', 14, 'TYPE BODY',
                       19, 'TABLE PARTITION', 20, 'INDEX PARTITION', 21, 'LOB',
                       22, 'LIBRARY', 23, 'DIRECTORY', 24, 'QUEUE',
                       28, 'JAVA SOURCE', 29, 'JAVA CLASS', 30, 'JAVA RESOURCE',
                       32, 'INDEXTYPE', 33, 'OPERATOR',
                       34, 'TABLE SUBPARTITION', 35, 'INDEX SUBPARTITION',
                       40, 'LOB PARTITION', 41, 'LOB SUBPARTITION',
                       42, NVL((SELECT 'REWRITE EQUIVALENCE'
                                FROM sum$ s
                                WHERE s.obj#=o.obj#
                                      and bitand(s.xpflags, 8388608) = 8388608),
                               'MATERIALIZED VIEW'),
                       43, 'DIMENSION',
                       44, 'CONTEXT', 46, 'RULE SET', 47, 'RESOURCE PLAN',
                       48, 'CONSUMER GROUP',
                       51, 'SUBSCRIPTION', 52, 'LOCATION',
                       55, 'XML SCHEMA', 56, 'JAVA DATA',
                       57, 'EDITION', 59, 'RULE',
                       60, 'CAPTURE', 61, 'APPLY',
                       62, 'EVALUATION CONTEXT',
                       66, 'JOB', 67, 'PROGRAM', 68, 'JOB CLASS', 69, 'WINDOW',
                       72, 'SCHEDULER GROUP', 74, 'SCHEDULE', 79, 'CHAIN',
                       81, 'FILE GROUP', 82, 'MINING MODEL', 87, 'ASSEMBLY',
```

```
                        90, 'CREDENTIAL', 92, 'CUBE DIMENSION', 93, 'CUBE',
                        94, 'MEASURE FOLDER', 95, 'CUBE BUILD PROCESS',
                        100, 'FILE WATCHER', 101, 'DESTINATION',
                       'UNDEFINED'),
       o.ctime, o.mtime,
       to_char(o.stime, 'YYYY-MM-DD:HH24:MI:SS'),
       decode(o.status, 0, 'N/A', 1, 'VALID', 'INVALID'),
       decode(bitand(o.flags, 2), 0, 'N', 2, 'Y', 'N'),
       decode(bitand(o.flags, 4), 0, 'N', 4, 'Y', 'N'),
       decode(bitand(o.flags, 16), 0, 'N', 16, 'Y', 'N'),
       o.namespace,
       o.defining_edition
from sys."_CURRENT_EDITION_OBJ" o, sys.user$ u
where o.owner# = u.user#
  and o.linkname is null
  and (o.type# not in (1  /* INDEX - handled below */,
                       10 /* NON-EXISTENT */)
       or
       (o.type# = 1 and 1 = (select 1
                               from sys.ind$ i
                              where i.obj# = o.obj#
                                and i.type# in (1, 2, 3, 4, 6, 7, 9))))
  and o.name != '_NEXT_OBJECT'
  and o.name != '_default_auditing_options_'
  and bitand(o.flags, 128) = 0
union all
select u.name, l.name, NULL, to_number(null), to_number(null),
       'DATABASE LINK',
       l.ctime, to_date(null), NULL, 'VALID','N','N', 'N', NULL, NULL
from sys.link$ l, sys.user$ u
where l.owner# = u.user#
```

DBA_SEGMENTS

段（segment）视图。

```
create or replace view DBA_SEGMENTS
    (OWNER, SEGMENT_NAME,
     PARTITION_NAME,
     SEGMENT_TYPE, SEGMENT_SUBTYPE,
     TABLESPACE_NAME,
     HEADER_FILE, HEADER_BLOCK,
     BYTES, BLOCKS, EXTENTS,
```

```
      INITIAL_EXTENT, NEXT_EXTENT,
      MIN_EXTENTS, MAX_EXTENTS, MAX_SIZE, RETENTION, MINRETENTION,
      PCT_INCREASE,
      FREELISTS, FREELIST_GROUPS,
      RELATIVE_FNO, BUFFER_POOL, FLASH_CACHE, CELL_FLASH_CACHE)
as
select owner, segment_name, partition_name, segment_type,
       segment_subtype, tablespace_name,
       header_file, header_block,
       decode(bitand(segment_flags, 131072), 131072, blocks,
           (decode(bitand(segment_flags,1),1,
            dbms_space_admin.segment_number_blocks(tablespace_id, relative_fno,
            header_block, segment_type_id, buffer_pool_id, segment_flags,
            segment_objd, blocks), blocks)))*blocksize,
       decode(bitand(segment_flags, 131072), 131072, blocks,
           (decode(bitand(segment_flags,1),1,
            dbms_space_admin.segment_number_blocks(tablespace_id, relative_fno,
            header_block, segment_type_id, buffer_pool_id, segment_flags,
            segment_objd, blocks), blocks))),
       decode(bitand(segment_flags, 131072), 131072, extents,
           (decode(bitand(segment_flags,1),1,
           dbms_space_admin.segment_number_extents(tablespace_id, relative_fno,
           header_block, segment_type_id, buffer_pool_id, segment_flags,
           segment_objd, extents) , extents))),
       initial_extent, next_extent, min_extents, max_extents, max_size,
       retention, minretention,
       pct_increase, freelists, freelist_groups, relative_fno,
       decode(buffer_pool_id, 1, 'KEEP', 2, 'RECYCLE', 'DEFAULT'),
       decode(flash_cache, 1, 'KEEP', 2, 'NONE', 'DEFAULT'),
       decode(cell_flash_cache, 1, 'KEEP', 2, 'NONE', 'DEFAULT')
from sys_dba_segs

--------------------------------------------------------------------------------
```

sys_dba_segs 也是一个视图，定义参考下一项

SYS_DBA_SEGS

段视图。

```
create or replace view sys.sys_dba_segs
(owner, segment_name, partition_name, segment_type, segment_type_id, segment_subtype,
tablespace_id, tablespace_name, blocksize, header_file, header_block, bytes, blocks, extents,
```

```
initial_extent, next_extent, min_extents, max_extents, max_size, retention, minretention,
pct_increase, freelists, freelist_groups, relative_fno, buffer_pool_id, flash_cache,
cell_flash_cache, segment_flags, segment_objd)
as
select NVL(u.name, 'SYS'), o.name, o.subname,
       so.object_type, s.type#,
       decode(bitand(s.spare1, 2097408), 2097152, 'SECUREFILE', 256, 'ASSM', 'MSSM'),
       ts.ts#, ts.name, ts.blocksize,
       f.file#, s.block#,
       s.blocks * ts.blocksize, s.blocks, s.extents,
       s.iniexts * ts.blocksize,
       s.extsize * ts.blocksize,
       s.minexts, s.maxexts,
       decode(bitand(s.spare1, 4194304), 4194304, bitmapranges, NULL),
       to_char(decode(bitand(s.spare1, 2097152), 2097152,
              decode(s.lists, 0, 'NONE', 1, 'AUTO', 2, 'MIN', 3, 'MAX',
                     4, 'DEFAULT', 'INVALID'), NULL)),
       decode(bitand(s.spare1, 2097152), 2097152, s.groups, NULL),
       decode(bitand(ts.flags, 3), 1, to_number(NULL),
                                      s.extpct),
       decode(bitand(ts.flags, 32), 32, to_number(NULL),
              decode(s.lists, 0, 1, s.lists)),
       decode(bitand(ts.flags, 32), 32, to_number(NULL),
              decode(s.groups, 0, 1, s.groups)),
       s.file#, bitand(s.cachehint, 3), bitand(s.cachehint, 12)/4,
       bitand(s.cachehint, 48)/16, NVL(s.spare1,0), o.dataobj#
from sys.user$ u, sys.obj$ o, sys.ts$ ts, sys.sys_objects so, sys.seg$ s,
     sys.file$ f
where s.file# = so.header_file
  and s.block# = so.header_block
  and s.ts# = so.ts_number
  and s.ts# = ts.ts#
  and o.obj# = so.object_id
  and o.owner# = u.user# (+)
  and s.type# = so.segment_type_id
  and o.type# = so.object_type_id
  and s.ts# = f.ts#
  and s.file# = f.relfile#
union all
select NVL(u.name, 'SYS'), un.name, NULL,
       decode(s.type#, 1, 'ROLLBACK', 10, 'TYPE2 UNDO'), s.type#,
       NULL, ts.ts#, ts.name, ts.blocksize, f.file#, s.block#,
       s.blocks * ts.blocksize, s.blocks, s.extents,
       s.iniexts * ts.blocksize, s.extsize * ts.blocksize, s.minexts,
```

```
       s.maxexts,
       decode(bitand(s.spare1, 4194304), 4194304, bitmapranges, NULL),
       NULL, NULL, s.extpct,
       decode(bitand(ts.flags, 32), 32, to_number(NULL),
              decode(s.lists, 0, 1, s.lists)),
       decode(bitand(ts.flags, 32), 32, to_number(NULL),
              decode(s.groups, 0, 1, s.groups)),
       s.file#, bitand(s.cachehint, 3), bitand(s.cachehint, 12)/4,
       bitand(s.cachehint, 48)/16, NVL(s.spare1,0), un.us#
from sys.user$ u, sys.ts$ ts, sys.undo$ un, sys.seg$ s, sys.file$ f
where s.file# = un.file#
  and s.block# = un.block#
  and s.ts# = un.ts#
  and s.ts# = ts.ts#
  and s.user# = u.user# (+)
  and s.type# in (1, 10)
  and un.status$ != 1
  and un.ts# = f.ts#
  and un.file# = f.relfile#
union all
select NVL(u.name, 'SYS'), to_char(f.file#) || '.' || to_char(s.block#), NULL,
       decode(s.type#, 2, 'DEFERRED ROLLBACK', 3, 'TEMPORARY',
                       4, 'CACHE', 9, 'SPACE HEADER', 'UNDEFINED'), s.type#,
       NULL, ts.ts#, ts.name, ts.blocksize,
       f.file#, s.block#,
       s.blocks * ts.blocksize, s.blocks, s.extents,
       s.iniexts * ts.blocksize,
       s.extsize * ts.blocksize,
       s.minexts, s.maxexts,
       decode(bitand(s.spare1, 4194304), 4194304, bitmapranges, NULL),
       NULL, NULL, decode(bitand(ts.flags, 3), 1, to_number(NULL),
                                     s.extpct),
       decode(bitand(ts.flags, 32), 32, to_number(NULL),
              decode(s.lists, 0, 1, s.lists)),
       decode(bitand(ts.flags, 32), 32, to_number(NULL),
              decode(s.groups, 0, 1, s.groups)),
       s.file#, bitand(s.cachehint, 3), bitand(s.cachehint, 12)/4,
       bitand(s.cachehint, 48)/16, NVL(s.spare1,0), s.hwmincr
from sys.user$ u, sys.ts$ ts, sys.seg$ s, sys.file$ f
where s.ts# = ts.ts#
  and s.user# = u.user# (+)
  and s.type# not in (1, 5, 6, 8, 10)
  and s.ts# = f.ts#
  and s.file# = f.relfile#
```

SYS_OBJECTS

```
create or replace view sys.sys_objects
(object_type, object_type_id, segment_type_id, object_id, header_file, header_block, ts_number)
as
select decode(bitand(t.property, 8192), 8192, 'NESTED TABLE', 'TABLE'), 2, 5,
       t.obj#, t.file#, t.block#, t.ts#
from sys.tab$ t
where bitand(t.property, 1024) = 0               /* exclude clustered tables */
union all
select 'TABLE PARTITION', 19, 5,
       tp.obj#, tp.file#, tp.block#, tp.ts#
from sys.tabpart$ tp
union all
select 'CLUSTER', 3, 5,
       c.obj#, c.file#, c.block#, c.ts#
from sys.clu$ c
union all
select decode(i.type#, 8, 'LOBINDEX', 'INDEX'), 1, 6,
       i.obj#, i.file#, i.block#, i.ts#
from sys.ind$ i
where i.type# in (1, 2, 3, 4, 6, 7, 8, 9)
union all
select 'INDEX PARTITION', 20, 6,
       ip.obj#, ip.file#, ip.block#, ip.ts#
from sys.indpart$ ip
union all
select 'LOBSEGMENT', 21, 8,
       l.lobj#, l.file#, l.block#, l.ts#
from sys.lob$ l
where (bitand(l.property, 64) = 0) or
      (bitand(l.property, 128) = 128)
union all
select 'TABLE SUBPARTITION', 34, 5,
       tsp.obj#, tsp.file#, tsp.block#, tsp.ts#
       from sys.tabsubpart$ tsp
union all
select 'INDEX SUBPARTITION', 35, 6,
       isp.obj#, isp.file#, isp.block#, isp.ts#
from sys.indsubpart$ isp
union all
select decode(lf.fragtype$, 'P', 'LOB PARTITION', 'LOB SUBPARTITION'),
```

```
       decode(lf.fragtype$, 'P', 40, 41), 8,
       lf.fragobj#, lf.file#, lf.block#, lf.ts#
from sys.lobfrag$ lf
```

DBA_TAB_COLS

表字段视图。

```
create or replace view DBA_TAB_COLS
    (OWNER, TABLE_NAME,
     COLUMN_NAME, DATA_TYPE, DATA_TYPE_MOD, DATA_TYPE_OWNER,
     DATA_LENGTH, DATA_PRECISION, DATA_SCALE, NULLABLE, COLUMN_ID,
     DEFAULT_LENGTH, DATA_DEFAULT, NUM_DISTINCT, LOW_VALUE, HIGH_VALUE,
     DENSITY, NUM_NULLS, NUM_BUCKETS, LAST_ANALYZED, SAMPLE_SIZE,
     CHARACTER_SET_NAME, CHAR_COL_DECL_LENGTH,
     GLOBAL_STATS, USER_STATS, AVG_COL_LEN, CHAR_LENGTH, CHAR_USED,
     V8O_FMT_IMAGE, DATA_UPGRADED, HIDDEN_COLUMN, VIRTUAL_COLUMN,
     SEGMENT_COLUMN_ID, INTERNAL_COLUMN_ID, HISTOGRAM, QUALIFIED_COL_NAME)
as
select u.name, o.name,
       c.name,
       decode(c.type#, 1, decode(c.charsetform, 2, 'NVARCHAR2', 'VARCHAR2'),
                       2, decode(c.scale, null,
                                 decode(c.precision#, null, 'NUMBER', 'FLOAT'),
                                 'NUMBER'),
                       8, 'LONG',
                       9, decode(c.charsetform, 2, 'NCHAR VARYING', 'VARCHAR'),
                       12, 'DATE',
                       23, 'RAW', 24, 'LONG RAW',
                       58, nvl2(ac.synobj#, (select o.name from obj$ o
                                 where o.obj#=ac.synobj#), ot.name),
                       69, 'ROWID',
                       96, decode(c.charsetform, 2, 'NCHAR', 'CHAR'),
                       100, 'BINARY_FLOAT',
                       101, 'BINARY_DOUBLE',
                       105, 'MLSLABEL',
                       106, 'MLSLABEL',
                       111, nvl2(ac.synobj#, (select o.name from obj$ o
                                 where o.obj#=ac.synobj#), ot.name),
                       112, decode(c.charsetform, 2, 'NCLOB', 'CLOB'),
                       113, 'BLOB', 114, 'BFILE', 115, 'CFILE',
                       121, nvl2(ac.synobj#, (select o.name from obj$ o
                                 where o.obj#=ac.synobj#), ot.name),
```

```
                    122, nvl2(ac.synobj#, (select o.name from obj$ o
                          where o.obj#=ac.synobj#), ot.name),
                    123, nvl2(ac.synobj#, (select o.name from obj$ o
                          where o.obj#=ac.synobj#), ot.name),
                    178, 'TIME(' ||c.scale|| ')',
                    179, 'TIME(' ||c.scale|| ')' || ' WITH TIME ZONE',
                    180, 'TIMESTAMP(' ||c.scale|| ')',
                    181, 'TIMESTAMP(' ||c.scale|| ')' || ' WITH TIME ZONE',
                    231, 'TIMESTAMP(' ||c.scale|| ')' || ' WITH LOCAL TIME ZONE',
                    182, 'INTERVAL YEAR(' ||c.precision#||') TO MONTH',
                    183, 'INTERVAL DAY(' ||c.precision#||') TO SECOND(' ||
                          c.scale || ')',
                    208, 'UROWID',
                    'UNDEFINED'),
      decode(c.type#, 111, 'REF'),
      nvl2(ac.synobj#, (select u.name from "_BASE_USER" u, obj$ o
                        where o.owner#=u.user# and o.obj#=ac.synobj#),
           ut.name),
      c.length, c.precision#, c.scale,
      decode(sign(c.null$),-1,'D', 0, 'Y', 'N'),
      decode(c.col#, 0, to_number(null), c.col#), c.deflength,
      c.default$, h.distcnt, h.lowval, h.hival, h.density, h.null_cnt,
      case when nvl(h.distcnt,0) = 0 then h.distcnt
           when h.row_cnt = 0 then 1
           when (h.bucket_cnt > 255
                 or
                 (h.bucket_cnt > h.distcnt
                  and h.row_cnt = h.distcnt
                  and h.density*h.bucket_cnt < 1))
              then h.row_cnt
           else h.bucket_cnt
      end,
      h.timestamp#, h.sample_size,
      decode(c.charsetform, 1, 'CHAR_CS',
                            2, 'NCHAR_CS',
                            3, NLS_CHARSET_NAME(c.charsetid),
                            4, 'ARG:'||c.charsetid),
      decode(c.charsetid, 0, to_number(NULL),
                          nls_charset_decl_len(c.length, c.charsetid)),
      decode(bitand(h.spare2, 2), 2, 'YES', 'NO'),
      decode(bitand(h.spare2, 1), 1, 'YES', 'NO'),
      h.avgcln,
      c.spare3,
      decode(c.type#, 1, decode(bitand(c.property, 8388608), 0, 'B', 'C'),
```

```
                        96, decode(bitand(c.property, 8388608), 0, 'B', 'C'),
                        null),
       decode(bitand(ac.flags, 128), 128, 'YES', 'NO'),
       decode(o.status, 1, decode(bitand(ac.flags, 256), 256, 'NO', 'YES'),
                          decode(bitand(ac.flags, 2), 2, 'NO',
                                 decode(bitand(ac.flags, 4), 4, 'NO',
                                        decode(bitand(ac.flags, 8), 8, 'NO',
                                               'N/A')))),
       decode(c.property, 0, 'NO', decode(bitand(c.property, 32), 32, 'YES',
                                          'NO')),
       decode(c.property, 0, 'NO', decode(bitand(c.property, 8), 8, 'YES',
                                          'NO')),
       decode(c.segcol#, 0, to_number(null), c.segcol#), c.intcol#,
       case when nvl(h.row_cnt,0) = 0 then 'NONE'
            when (h.bucket_cnt > 255
                  or
                  (h.bucket_cnt > h.distcnt and h.row_cnt = h.distcnt
                   and h.density*h.bucket_cnt < 1))
                then 'FREQUENCY'
            else 'HEIGHT BALANCED'
       end,
       decode(bitand(c.property, 1024), 1024,
              (select decode(bitand(cl.property, 1), 1, rc.name, cl.name)
               from sys.col$ cl, attrcol$ rc where cl.intcol# = c.intcol#-1
               and cl.obj# = c.obj# and c.obj# = rc.obj#(+) and
               cl.intcol# = rc.intcol#(+)),
              decode(bitand(c.property, 1), 0, c.name,
                     (select tc.name from sys.attrcol$ tc
                      where c.obj# = tc.obj# and c.intcol# = tc.intcol#)))
from sys.col$ c, sys."_CURRENT_EDITION_OBJ" o, sys.hist_head$ h, sys.user$ u,
     sys.coltype$ ac, sys.obj$ ot, sys."_BASE_USER" ut
where o.obj# = c.obj#
  and o.owner# = u.user#
  and c.obj# = h.obj#(+) and c.intcol# = h.intcol#(+)
  and c.obj# = ac.obj#(+) and c.intcol# = ac.intcol#(+)
  and ac.toid = ot.oid$(+)
  and ot.type#(+) = 13
  and ot.owner# = ut.user#(+)
  and (o.type# in (3, 4)                                    /* cluster, view */
       or
       (o.type# = 2     /* tables, excluding iot - overflow and nested tables */
        and
        not exists (select null
                    from sys.tab$ t
```

```
                        where t.obj# = o.obj#
                          and (bitand(t.property, 512) = 512 or
                               bitand(t.property, 8192) = 8192))))
```

DBA_TABLES

表（table）视图。

```
create or replace view DBA_TABLES
    (OWNER, TABLE_NAME, TABLESPACE_NAME, CLUSTER_NAME, IOT_NAME, STATUS,
     PCT_FREE, PCT_USED,
     INI_TRANS, MAX_TRANS,
     INITIAL_EXTENT, NEXT_EXTENT,
     MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE,
     FREELISTS, FREELIST_GROUPS, LOGGING,
     BACKED_UP, NUM_ROWS, BLOCKS, EMPTY_BLOCKS,
     AVG_SPACE, CHAIN_CNT, AVG_ROW_LEN,
     AVG_SPACE_FREELIST_BLOCKS, NUM_FREELIST_BLOCKS,
     DEGREE, INSTANCES, CACHE, TABLE_LOCK,
     SAMPLE_SIZE, LAST_ANALYZED, PARTITIONED,
     IOT_TYPE, TEMPORARY, SECONDARY, NESTED,
     BUFFER_POOL, FLASH_CACHE,
     CELL_FLASH_CACHE, ROW_MOVEMENT,
     GLOBAL_STATS, USER_STATS, DURATION, SKIP_CORRUPT, MONITORING,
     CLUSTER_OWNER, DEPENDENCIES, COMPRESSION, COMPRESS_FOR,DROPPED, READ_ONLY,
     SEGMENT_CREATED,RESULT_CACHE)
as
select u.name, o.name,
       decode(bitand(t.property,2151678048), 0, ts.name,
              decode(t.ts#, 0, null, ts.name)),
       decode(bitand(t.property, 1024), 0, null, co.name),
       decode((bitand(t.property, 512)+bitand(t.flags, 536870912)),
              0, null, co.name),
       decode(bitand(t.trigflag, 1073741824), 1073741824, 'UNUSABLE', 'VALID'),
       decode(bitand(t.property, 32+64), 0, mod(t.pctfree$, 100), 64, 0, null),
       decode(bitand(ts.flags, 32), 32, to_number(NULL),
          decode(bitand(t.property, 32+64), 0, t.pctused$, 64, 0, null)),
       decode(bitand(t.property, 32), 0, t.initrans, null),
       decode(bitand(t.property, 32), 0, t.maxtrans, null),
       decode(bitand(t.property, 17179869184), 17179869184,
                    ds.initial_stg * ts.blocksize,
                    s.iniexts * ts.blocksize),
       decode(bitand(t.property, 17179869184), 17179869184,
```

```
        ds.next_stg * ts.blocksize,
        s.extsize * ts.blocksize),
    decode(bitand(t.property, 17179869184), 17179869184,
        ds.minext_stg, s.minexts),
    decode(bitand(t.property, 17179869184), 17179869184,
        ds.maxext_stg, s.maxexts),
    decode(bitand(ts.flags, 3), 1, to_number(NULL),
        decode(bitand(t.property, 17179869184), 17179869184,
                ds.pctinc_stg, s.extpct)),
    decode(bitand(ts.flags, 32), 32, to_number(NULL),
      decode(bitand(o.flags, 2), 2, 1,
          decode(bitand(t.property, 17179869184), 17179869184,
              ds.frlins_stg, decode(s.lists, 0, 1, s.lists)))),
    decode(bitand(ts.flags, 32), 32, to_number(NULL),
      decode(bitand(o.flags, 2), 2, 1,
          decode(bitand(t.property, 17179869184), 17179869184,
              ds.maxins_stg, decode(s.groups, 0, 1, s.groups)))),
    decode(bitand(t.property, 32+64), 0,
          decode(bitand(t.flags, 32), 0, 'YES', 'NO'), null),
    decode(bitand(t.flags,1), 0, 'Y', 1, 'N', '?'),
    t.rowcnt,
    decode(bitand(t.property, 64), 0, t.blkcnt, null),
    decode(bitand(t.property, 64), 0, t.empcnt, null),
    t.avgspc, t.chncnt, t.avgrln, t.avgspc_flb,
    decode(bitand(t.property, 64), 0, t.flbcnt, null),
    lpad(decode(t.degree, 32767, 'DEFAULT', nvl(t.degree,1)),10),
    lpad(decode(t.instances, 32767, 'DEFAULT', nvl(t.instances,1)),10),
    lpad(decode(bitand(t.flags, 8), 8, 'Y', 'N'),5),
    decode(bitand(t.flags, 6), 0, 'ENABLED', 'DISABLED'),
    t.samplesize, t.analyzetime,
    decode(bitand(t.property, 32), 32, 'YES', 'NO'),
    decode(bitand(t.property, 64), 64, 'IOT',
          decode(bitand(t.property, 512), 512, 'IOT_OVERFLOW',
          decode(bitand(t.flags, 536870912), 536870912, 'IOT_MAPPING', null))),
    decode(bitand(o.flags, 2), 0, 'N', 2, 'Y', 'N'),
    decode(bitand(o.flags, 16), 0, 'N', 16, 'Y', 'N'),
    decode(bitand(t.property, 8192), 8192, 'YES',
        decode(bitand(t.property, 1), 0, 'NO', 'YES')),
    decode(bitand(o.flags, 2), 2, 'DEFAULT',
        decode(bitand(decode(bitand(t.property, 17179869184), 17179869184,
                ds.bfp_stg, s.cachehint), 3),
                1, 'KEEP', 2, 'RECYCLE', 'DEFAULT')),
    decode(bitand(o.flags, 2), 2, 'DEFAULT',
        decode(bitand(decode(bitand(t.property, 17179869184), 17179869184,
```

```
                        ds.bfp_stg, s.cachehint), 12)/4,
                        1, 'KEEP', 2, 'NONE', 'DEFAULT')),
       decode(bitand(o.flags, 2), 2, 'DEFAULT',
              decode(bitand(decode(bitand(t.property, 17179869184), 17179869184,
                        ds.bfp_stg, s.cachehint), 48)/16,
                        1, 'KEEP', 2, 'NONE', 'DEFAULT')),
       decode(bitand(t.flags, 131072), 131072, 'ENABLED', 'DISABLED'),
       decode(bitand(t.flags, 512), 0, 'NO', 'YES'),
       decode(bitand(t.flags, 256), 0, 'NO', 'YES'),
       decode(bitand(o.flags, 2), 0, NULL,
          decode(bitand(t.property, 8388608), 8388608,
                 'SYS$SESSION', 'SYS$TRANSACTION')),
       decode(bitand(t.flags, 1024), 1024, 'ENABLED', 'DISABLED'),
       decode(bitand(o.flags, 2), 2, 'NO',
           decode(bitand(t.property, 2147483648), 2147483648, 'NO',
              decode(ksppcv.ksppstvl, 'TRUE', 'YES', 'NO'))),
       decode(bitand(t.property, 1024), 0, null, cu.name),
       decode(bitand(t.flags, 8388608), 8388608, 'ENABLED', 'DISABLED'),
       case when (bitand(t.property, 32) = 32) then
         null
       when (bitand(t.property, 17179869184) = 17179869184) then
          decode(bitand(ds.flags_stg, 4), 4, 'ENABLED', 'DISABLED')
       else
         decode(bitand(s.spare1, 2048), 2048, 'ENABLED', 'DISABLED')
       end,
       case when (bitand(t.property, 32) = 32) then
         null
       when (bitand(t.property, 17179869184) = 17179869184) then
          decode(bitand(ds.flags_stg, 4), 4,
          case when bitand(ds.cmpflag_stg, 3) = 1 then 'BASIC'
               when bitand(ds.cmpflag_stg, 3) = 2 then 'OLTP'
               else decode(ds.cmplvl_stg, 1, 'QUERY LOW',
                                          2, 'QUERY HIGH',
                                          3, 'ARCHIVE LOW',
                                             'ARCHIVE HIGH') end,
               null)
       else
         decode(bitand(s.spare1, 2048), 0, null,
         case when bitand(s.spare1, 16777216) = 16777216
                   then 'OLTP'
              when bitand(s.spare1, 100663296) = 33554432  -- 0x2000000
                   then 'QUERY LOW'
              when bitand(s.spare1, 100663296) = 67108864  -- 0x4000000
                   then 'QUERY HIGH'
```

```
               when bitand(s.spare1, 100663296) = 100663296 -- 0x2000000+0x4000000
                    then 'ARCHIVE LOW'
               when bitand(s.spare1, 134217728) = 134217728 -- 0x8000000
                    then 'ARCHIVE HIGH'
               else 'BASIC' end)
       end,
       decode(bitand(o.flags, 128), 128, 'YES', 'NO'),
       decode(bitand(t.trigflag, 2097152), 2097152, 'YES', 'NO'),
       decode(bitand(t.property, 17179869184), 17179869184, 'NO',
              decode(bitand(t.property, 32), 32, 'N/A', 'YES')),
       decode(bitand(t.property,16492674416640),2199023255552,'FORCE',
                 4398046511104,'MANUAL','DEFAULT')
from sys.user$ u, sys.ts$ ts, sys.seg$ s, sys.obj$ co, sys.tab$ t, sys.obj$ o,
     sys.obj$ cx, sys.user$ cu, x$ksppcv ksppcv, x$ksppi ksppi,
     sys.deferred_stg$ ds
where o.owner# = u.user#
  and o.obj# = t.obj#
  and bitand(t.property, 1) = 0
  and bitand(o.flags, 128) = 0
  and t.bobj# = co.obj# (+)
  and t.ts# = ts.ts#
  and t.obj# = ds.obj# (+)
  and t.file# = s.file# (+)
  and t.block# = s.block# (+)
  and t.ts# = s.ts# (+)
  and t.dataobj# = cx.obj# (+)
  and cx.owner# = cu.user# (+)
  and ksppi.indx = ksppcv.indx
  and ksppi.ksppinm = '_dml_monitoring_enabled'
```

DBA_TABLESPACES

表空间视图。

```
create or replace view DBA_TABLESPACES
    (TABLESPACE_NAME, BLOCK_SIZE, INITIAL_EXTENT, NEXT_EXTENT, MIN_EXTENTS,
     MAX_EXTENTS, MAX_SIZE, PCT_INCREASE, MIN_EXTLEN,
     STATUS, CONTENTS, LOGGING, FORCE_LOGGING, EXTENT_MANAGEMENT,
     ALLOCATION_TYPE, PLUGGED_IN,
     SEGMENT_SPACE_MANAGEMENT, DEF_TAB_COMPRESSION, RETENTION, BIGFILE,
     PREDICATE_EVALUATION, ENCRYPTED, COMPRESS_FOR)
as select ts.name, ts.blocksize, ts.blocksize * ts.dflinit,
          decode(bitand(ts.flags, 3), 1, to_number(NULL),
```

```
                ts.blocksize * ts.dflincr),
        ts.dflminext,
        decode(ts.contents$, 1, to_number(NULL), ts.dflmaxext),
        decode(bitand(ts.flags, 4096), 4096, ts.affstrength, NULL),
        decode(bitand(ts.flags, 3), 1, to_number(NULL), ts.dflextpct),
        ts.blocksize * ts.dflminlen,
        decode(ts.online$, 1, 'ONLINE', 2, 'OFFLINE',
               4, 'READ ONLY', 'UNDEFINED'),
        decode(ts.contents$, 0, (decode(bitand(ts.flags, 16), 16, 'UNDO',
               'PERMANENT')), 1, 'TEMPORARY'),
        decode(bitand(ts.dflogging, 1), 0, 'NOLOGGING', 1, 'LOGGING'),
        decode(bitand(ts.dflogging, 2), 0, 'NO', 2, 'YES'),
        decode(ts.bitmapped, 0, 'DICTIONARY', 'LOCAL'),
        decode(bitand(ts.flags, 3), 0, 'USER', 1, 'SYSTEM', 2, 'UNIFORM',
               'UNDEFINED'),
        decode(ts.plugged, 0, 'NO', 'YES'),
        decode(bitand(ts.flags,32), 32,'AUTO', 'MANUAL'),
        decode(bitand(ts.flags,64), 64,'ENABLED', 'DISABLED'),
        decode(bitand(ts.flags,16), 16, (decode(bitand(ts.flags, 512), 512,
               'GUARANTEE', 'NOGUARANTEE')), 'NOT APPLY'),
        decode(bitand(ts.flags,256), 256, 'YES', 'NO'),
        decode(tsattr.storattr, 1, 'STORAGE', 'HOST'),
        decode(bitand(ts.flags,16384), 16384, 'YES', 'NO'),
        decode(bitand(ts.flags,64), 0, null,
          (case when bitand(ts.flags,  65536) = 65536
                  then 'OLTP'
                when bitand(ts.flags, (131072+262144)) = 131072
                  then 'QUERY LOW'
                when bitand(ts.flags, (131072+262144)) = 262144
                  then 'QUERY HIGH'
                when bitand(ts.flags, (131072+262144)) = (131072+262144)
                  then 'ARCHIVE LOW'
                when bitand(ts.flags, 524288) = 524288
                  then 'ARCHIVE HIGH'
                else 'BASIC' end))
from sys.ts$ ts, sys.x$kcfistsa tsattr
where ts.online$ != 3
and bitand(flags,2048) != 2048
and ts.ts# = tsattr.tsid
```

总结一下

看到上面这么多的信息，你一定觉得很乱，下面我们梳理一下，筛选出我们需要的部分。我们先确定一下，都要导出哪些数据，限定了范围，就能有针对性的分析问题了。第一，我们要导出普通表的数据，第二，要导出分区表的数据，包括分区和子分区，第三，要导出含有 LOB 字段的表的 LOB 数据，第四，还要导出 cluster 表的数据，在导出数据字典信息时会遇到 cluster 表。其他类型的表我们就不考虑了，如果你有兴趣，可以自己研究一下，也是长知识的好机会。导出的字段类型我们也限定一下，只支持 Oralce 内建的数据类型，不支持用户定义类型（UDT）和其他对象（Object）类型。对于压缩表和加密字段的表，我们也不打算支持，压缩表对应着段的压缩，由 seg$的 spare1 字段的 0x800 标记。

在另一篇分析 Oracle 存储结构的文档中我们看到，要导出数据，关键在于找到数据段对应的文件号（HEADER_FILE）和块号（HEADER_BLOCK），找到这个 DBA，就找到了段头的管理块，就能找到段包含的所有 extents，那么也就能导出这个段的全部数据。

…… 部分信息省略 ……

前面说到了在 Oracle 数据库打开的情况下怎样处理数据字典，但是我们的程序可能是在数据库不能启动的情况下使用，这时该怎样获取数据字典呢？原理还是一样的，我们要想办法把上面提到的基础表的数据先导出来，构建出数据字典。要导出基础表的数据就要知道基础表的段头地址，而段头地址又是从数据字典的基础表中获得，这一来似乎进入死循环了。那么 Oracle 怎样处理这种情况呢？这要从 Oracle 的启动过程说起，原来，Oracle 在刚启动时，在内存中创建了一些基础表，这些基础表段头有固定的地址，这样 Oracle 就可以加载数据字典了，其他的字典再从这些基础表获得，这样就从最基础的几个表把数据字典一层层构建出来，我们也借用这种方法。最开始的一些基础表的信息在什么地方呢？Oracle 在最开始创建的内存表叫 bootstrap$,里面保存着最开始的基础表建表语句，Oracle 拿到这些语句后逐条执行，创建出其他的内存表，我们如果拿到这些语句，就能知道一些基础表的段头地址。那么 bootstrap$的地址在哪儿呢？这个地址叫 root dba，也是段头地址，保存在 1 号数据文件，也就是 SYSTEM

表空间的第一个数据文件的 block 1 中，你可以回头看看另一篇文档中的内容。好了，有了上面我们分析的信息，数据字典的获取应该不是问题了。

你也许进一步会问，既然在非正常情况下才用到 DUL，那么如果 SYSTEM 表空间的数据文件也损坏了，那怎么办呢？真是个棘手的问题，如果出现这种情况，那你能完全找回数据的机会就很小了，但优秀的软件也要尽最大可能把数据导出来，我们不打算处理这种情况，只是提供一点思路，如果你有兴趣可以更深入研究下去，那你会变成一个很棒的人。如果 SYSTEM 表空间的文件只是部分损坏，并且不能从 bootstrap$拿到信息，那就扫描整个系统表空间文件吧，把所有段头块都找出来，段头块的信息中不是有 dataobj#吗？这个是数据层的 obj#，如果是数据字典的基表，这个 dataobj#和 obj#的值是一样的，Oracle 基表的 obj# 也都是固定的，如果是几个基表组成的 cluster，那么 dataobj#会是 cluster 的 dataobj#，这些信息也是固定的，tab#也是固定的，从这里开始，就能构建出数据字典。极端的情况，SYSTEM 表空间的数据文件全丢了，那怎么办呢？数据字典是不能复原了，但还是用同样的方法，把所有数据文件都扫描一遍，把段头的块都找出来，然后到每个段的数据块中找到数据，扫描一些数据，然后根据 Oracle 的数据类型，把字段信息尽量还原出来，Oracle 的一些数据类型长度是固定的（比如 DATE，TIMESTAMP 系列，INTERVAL 系列，BINARY_FLOAT，BINARY_DOUBLE 等），CHAR 类型和 LOB 类型也有明显的特征，VARCHAR2 和 RAW 不太好区分，遇到这样的情况可能需要人为去判断一下了，主要还是为了还原出字段的信息，这是数据导出中最重要的部分。

数据结构设计

在准备写程序之前，要把用到的数据结构大致设计一下，这时程序该怎样写应该在脑子里有一个大概的轮廓了，先把数据结构中必要的元素罗列出来，不够的在写程序的过程中补充。我们在这儿用 C 语言的格式定义结构体。

数据字典结构

```
/* user$ 表数据结构 */
typedef struct __user
```

```
{
  uint32_t      uid;            /* 用户 ID */
  uint32_t      status;         /* 用户状态 */
  uint32_t      dts;            /* 缺省表空间号 */
  uint32_t      tts;            /* 临时表空间号 */
  char          name[32];       /* 用户名称 */
} USER;

/* ts$ 表数据结构 */
typedef struct __tablespace
{
  uint32_t      tsn;            /* 表空间号 */
  uint32_t      owner;          /* 属主 ID */
  char          name[32];       /* 表空间名称 */
} TS;

/* file$ 表数据结构 */
typedef struct __datafile
{
  uint32_t      fno;            /* 文件号 */
  uint32_t      rfn;            /* 相对文件号 */
  uint32_t      tsn;            /* 表空间号 */
  uint32_t      blks;           /* 包含的块个数 */
  char          name[1024];     /* 全路径文件名称 */
} DATAFILE;

/* obj$ 表数据结构 */
typedef struct __object
{
  uint32_t      flag;           /* 标志 */
  uint32_t      objn;           /* 对象 ID */
  uint32_t      owner;          /* 属主 ID */
  uint32_t      type;           /* 对象类型 */
  char          name[32];       /* 对象名称 */
} OBJ;

/* tab$ 表数据结构 */
typedef struct __table
{
  uint32_t      flag;           /* 标志 */
  uint32_t      objn;           /* 对象 ID */
  uint32_t      objd;           /* 段内对象 ID */
  uint32_t      tabn;           /* cluster 中表的编号 */
  uint32_t      ncol;           /* 字段个数 */
```

```
  uint32_t      ui;             /* 对应的用户数组索引下标 */
  uint32_t      oi;             /* 对应的对象数组索引下标 */
  uint32_t      coff;           /* 字段信息数据的偏移量 */
  uint32_t      fno;            /* 段头的文件号 */
  uint32_t      blk;            /* 段头的块号 */
} TAB;

/* col$ 表数据结构 */
typedef struct __column
{
  uint32_t      flag;           /* 标志 */
  uint32_t      cid;            /* 字段 ID */
  uint32_t      scid;           /* 段内字段 ID */
  uint32_t      ctyp;           /* 字段数据类型 */
  uint32_t      clen;           /* 字段数据长度，以字节为单位 */
  uint32_t      rsv;            /* 保留 */
  char          name[32];       /* 字段名称 */
} COL;
```

…… 部分信息省略 ……

存储结构

…… 部分信息省略 ……

配置参数

此类软件都有若干个配置文件，我们在这儿简化一下配置，也留下以后扩充的余地。先设置一个主配置文件，这是所有全局参数和其他配置文件的入口，配置参数写入到这个文件中，参数的格式采用 ini 文件格式，就是 pname=pvalue 的方式，注释为#开头。主配置文件命名为 config.ini，再命名一个数据文件列表配置，叫做 dbfile.list，里面存放 Oracle 数据文件的全路径名称。举例如下：

config.ini 内容：

```
dictdir=./dict
datadir=./data
```

```
datafiles=dbfiles.list
```

dbfiles.list 内容：

```
/oracle11/oradata/ora11g/system01.dbf
/oracle11/oradata/ora11g/users01.dbf
/oracle11/product/11.2.0/dbs/road_ts01.dbf
```

存储文件设计

数据字典存储

数据字典存储采用结构化存储的方法，按照前面定义的结构存储，每个字典表一个文件的方式。数据字典有导出和加载两个命令，导出时从数据文件中把字典信息写入到存储文件，加载时从数据字典存储文件读入到内存中。

数据存储

数据的导出可以按用户或按一张表的方式导出，我们把一次导出的数据存放到一个文件中，文件名称在导出用户时用 USERNAME.dat 表示，导出一张表时，用 USERNAME_TABLENAME.dat 表示。存储结构采用自己的设计，数据的装载使用自己的程序去完成。为了导出的数据能够在不同硬件架构的机器中装载，我们存储的数据文件中只要有整数的变量，都采用大字节序的方式存储。

存储文件分三部分构成，第一部分是文件头，包含导出的用户信息，字符集信息，表的个数等。第二部分是表信息，包含结构化的组合好的表信息，包含表名称，标志，字段个数，数据的偏移位置等。第三部分是表的数据，包含字段的信息结构，和一行行的数据信息。

存储文件中最重要的是表的行数据结构。表数据开始前还有一段是表的字段数据字典，这里要包含字段的类型，字段最大长度，字段名称。接下来就是表中的每一行数据，从 Oracle 的字段类型分析中我们知道除了大字段类型，其他数据类型最大长度都小于 4000 字节，由于两个字节的无符号整数最大能表示 65535 个字节，所以用两个字节存储数据长度就足够了，所以每个字段的结构用数据长

度（2 字节）后面紧跟数据的方式就能表示。在数据中还有几个特殊情况，第一个，如果字段数据为空（NULL），那么只要一个长度就可以了，首先想到的是用 0x0000 两个字节来表示空数据，但是我们还需要在这两个字节中预留一些特殊的值来代表其他内容，所以还不能这么来用。我们先看一下要预留些什么值来代表什么含义，先从大的方面考虑，如果一个表的数据结束了，我们怎么知道这种情况呢？先预留一个值 0xFFFF，这个值作为一个表结束的标志，在遇到数据长度为 0xFFFF 时，就知道这个表的全部数据处理完了。再一个，如果一行数据结束了，我们怎么知道呢？再预留一个值 0x0000，这个值作为一行数据结束的标志，遇到它时，就知道该处理下一行数据了。除掉这两个预留的值，我们用 0xFFFE 来表示空数据吧。前面在讨论字段类型时，看到还有两类大字段类型，一个为 LOB 类型，一个为 LONG 类型，这两个类型都要进行分片操作，我们把分片的数据放在一行后面，在行内用一个值来代表分片字段，0xFFFC 表示 LOB 字段，0xFFFB 表示 LONG 字段。分片数据以 0xFFFD 开始，以 0x0000 结束，中间是数据片段长度（2 字节）和数据片段的组合，数据片段长度我们取最大 32K（32768 字节）。

通过以上描述，数据存储结构已经清楚了。下面我们通过几个示意图把这些结构再明确一下。

文件头信息结构

导出程序名称（字符，32字节）
导出表的属主名称（字符，32字节）
导出数据字符集环境变量（字符，32字节）
表信息偏移量（数字，8字节）
表数据偏移量（数字，8字节）
导出表的个数（数字，4字节）

表信息结构

表的名称（字符，32字节）
表的标志（数字，4字节）
表的字段个数（数字，4字节）
表数据偏移量（数字，8字节）

表字段信息结构

字段名称（字符，32字节）
字段标志（数字，4字节）
字段类型（数字，4字节）
字段最大长度（数字，4字节）

表内一行数据的存储结构

字段1长度（数字，2字节）	字段1数据	……	字段n长度（数字，2字节）	字段n数据

字段长度 = 0x0000，一行数据结束
字段长度 = 0xFFFF，整个表的数据结束
字段长度 = 0xFFFE，字段值为空（NULL）
字段长度 = 0xFFFC，字段是LOB数据
字段长度 = 0xFFFB，字段是LONG数据
字段长度 = 0xFFFD，分片数据开始标志

分片数据的存储结构

分片起始标志（0xFFFD）	分片1长度（数字，2字节）	分片1数据	……	分片n长度（数字，2字节）	分片n数据	分片结束标志（0x0000）

程序结构设计

到了这里我们需要给软件取一个名字了，为了方便并能体现软件功能，我们叫它 mydul，以后的描述中提到 mydul 即代表软件本身。

命令设计

mydul 在启动时指定参数名称，如果不指定，采用默认参数文件 config.ini，如果指定，采用 mydul config=config_filename 的方式。mydul 启动后，采用命令驱动的运行方式，mydul 显示一个命令提示符，等待用户输入命令，mydul

根据用户命令，执行下一步操作，然后返回提示符状态，等待下次命令。下面我们来设计一下 mydul 的命令。

命令名称	命令参数	命令描述
export	dict	导出数据字典，mydul 启动后，使用本命令从数据文件中导出数据字典，存在本地
load	dict	加载数据字典，如果已经导出过数据字典，下次启动时，直接加载数据字典就可以，不用每次都导出
unload	user <username>	导出某用户下所有表的数据，每次只能导出一个用户下的数据，如果用户名称区分大小写，使用引号标识，如："UserName"
unload	table <user.table>	导出某用户一张表的数据，如果表名区分大小写，使用引号标识，如："TableName"
set	user <username>	设置当前用户，设置用户后，在命令中可以不用再指定用户，命令使用默认的当前用户。
show	user	显示当前用户
list	files	显示数据文件的信息
list	users	显示数据字典中的用户信息
list	tables <user>	显示数据字典中表的信息，如果显示某个用户下的表，后面跟用户名称。例如显示用户 TOM 下的表，用 list tables TOM 命令
list	objects <user>	显示数据字典中对象信息，用户规则同上
list	parts <user.table>	显示数据字典中分区的信息，如果显示某个表的分区信息，例如显示用户 TOM 下的表 Custom，用 list parts TOM."Custom"
desc	<user.table>	显示一个表的字段信息。
help		显示帮助信息
exit		退出程序

主程序设计

```
int main()
{
  /* 解析命令行参数 */
  parse_arguments();

  /* 加载配置文件 */
  load_config();

  /* 分配工作内存 */
  malloc();
```

```
  /* 加载 Oracle 数据文件 */
  load_datafile();

  /* 打开文件，分析出文件号，相对文件号，表空间号等信息 */
  open_datafile();

  /* 进入主循环，等待用户输入命令 */
  while (1)
  {
    /* 等待用户输入命令 */
    scanf();

    /* 去除命令前后多余的空格、换行等字符 */

    /* 判断命令，执行命令相应的处理函数 */

    /* 如果是退出命令，退出循环，结束程序 */
  }

  /* 关闭数据文件 */
  /* 释放数据字典占用的内存 */
  /* 释放工作内存 */

  return (0);
}
```

其他程序设计

从主程序的流程看到，处理逻辑比较简单，也很清晰。首先要导出数据字典，这一步从 Oracle 系统表空间数据文件中把我们用到的数据字典信息抓出来，写到我们自己定义的文件中，在导出后还要对字典中的数据按照一定的选项排序。然后把数据字典信息读入到内存中，在这一步中还要把字典中一些对象关系设置好，比如表怎么跟对象关联的，对象怎样跟用户关联的，字段怎样跟表关联的，分区和子分区怎样跟基础表关联的等等。最后一步就是把表的数据卸载(unload)出来，这一步中要依靠数据字典的帮助，找到数据在哪些段区（extent）内，怎样解释数据等等。

我们不再一一来设计这些程序结构，只拿出几个典型的函数出来设计一下。

剩下的工作在写代码时完成，在编码的过程中可能还要考虑到一些处理的技巧，那可能只有在写到那段代码时才有的灵感，在这儿是不能设计的。

处理数据字典

```
int export_dict_handle()
{
  /* 打开系统表空间的第一个数据文件时（文件号为 1），得到 rootdba */

  /* rootdba 是 bootstrap$表的段头块地址，读出这个块数据 */

  /* 根据不同的段头类型，得到 extent map 头地址和 extent map 数组地址 */

  /* 循环读取 extent 的信息 */
  for (i=0; i<extent 个数; i++)
  {
    /* 读取这个 extent 的所有块数据 */
    /* 依次处理每一个块中的 bootstrap$表的信息 */
  }

  /* 如果还有下一个 extent map，继续处理其中的 extent */

  /* 下面处理 bootstrap$表中包含的字典基础表的内容 */

  /* 导出用户 user$ 表数据 */
  export_user();

  /* 导出表空间 ts$ 表数据 */
  export_ts();

  /* 导出数据库对象 obj$ 表数据 */
  export_obj();

  /* 导出数据库表 tab$ 表数据 */
  export_tab();

  /* 导出聚簇 clu$ 表的数据 */
  export_clu();

  /* 导出列 col$ 表的数据 */
  export_col();
```

```
  /* 导出 lob 索引 ind$ 表的数据 */
  export_lob_ind();

  /* 对导出的字典表数据排序 */

  /* 在排序对象表 obj$ 时把其他不在 bootstrap$中的对象 ID 找到
   * 得到 lob$，tabpart$，tabsubpart$，tabcompart$，props$的 obj#
   * 在排序数据库表 tab$ 时把上面表的 rdba 得到
   */

  /* 导出 lob$ 表的数据 */
  export_lob();

  /* 导出分区 tabpart$ 表的数据 */
  export_tabpart();

  /* 导出子分区 tabsubpart$ 表的数据 */
  export_tabsubpart();

  /* 导出组合分区 tabcompart$ 表的数据 */
  export_tabcompart();

  /* 导出 props$ 表的数据 */
  export_props();

  /* 对上面导出的字典表数据排序 */

  return (0);
}
```

…… 部分信息省略 ……

处理普通表

…… 部分信息省略 ……

处理行连接

…… 部分信息省略 ……

处理 BasicFile LOB

…… 部分信息省略 ……

处理 SecureFile LOB

…… 部分信息省略 ……

开始编码

作为一个程序员，编码工作是最令人兴奋的一个过程，你心中的想法马上就要通过代码实现了。通过上面的设计或者说是思考，你一定会有一个写作的思路，甚至有一些逻辑已经在脑中放映了很多遍了，你一定会迫不及待的把它表达出来。编写代码也是一个再设计的过程，一些数据结构是否合理，结构中是否还要添加内容啊，这些都需要在写代码时决定，如果觉得有一个函数的代码写得不好，或者没法写下去，那么还要停下来，把这部分的逻辑仔细考虑清楚，然后再动手。总之，写代码是一个反复的过程，如果思路不清楚，写出来的逻辑结构也不会好，所以不要偷懒，以为一切问题到了写代码时就完全解决了，最好在写某个函数之前，先在脑子里过一遍，如果自己写完的代码自己都不想看，那还能指望别人看明白吗？

编写一个 makefile 文件，把写完的代码源文件加进去，写完一个编译一次，先把语法错误解决掉。

写代码时可以先把大的逻辑写一遍，把一些相关性低的逻辑抽象成函数，函数可以先空着，把函数要实现的功能描述清楚，等后面再写。大的逻辑写完后你的程序基本框架就完整了，可以先把空的函数，写一些打印语句，输出一些函数功能，这样编译后就可以先跑一下，然后就开始把那些空白函数添上代码吧。

调试

代码写完了，在一般的情况下不会顺利的跑起来，要不是内存段错误

（segment fault），就是运行的结果与期望的相差甚远。不用担心，代码是人写出来的，出现问题是正常的，关键是怎样发现问题，这要靠调试程序来解决。

把你的程序划分成几个大块，在一些地方划分节点，打印一些调试信息，这样便于确定问题出在哪一部分。确定了问题在哪一块，就可以再增加一些打印信息，把问题确定在一小段代码范围内，仔细检查一下这段代码，如果能找到问题，修改后再跑，看问题是否还在。或者直接用 gdb 去调试，把断点定位在出问题的的代码前，逐条执行去看看结果，这样很容易就能找出问题。

在这里列举几个调试中发现的问题，也可以回顾一下当时的场景。

…… 部分信息省略 ……

从整个调试的过程来看，有很多都是小的手误，这可以通过仔细查看程序就能解决掉，也有隐藏的比较深的，需要 gdb 调试才能找出来。还有一部分是由于对 Oracle 数据块结构的认识不深入造成的，可以通过调试程序的过程加深理解，也可能会重新做实验来检验以前的理解，这部分就成了积累的知识。还有一部分是逻辑考虑不周全造成的，程序不是一两天就能写成的，在编写代码的过程中会考虑很多东西，也会反复斟酌处理的逻辑，有很多是已经想到了，但在写程序时又忘记了，所以最好是在考虑到处理逻辑有不完善的地方就立即记下来，在写到那部分程序时对照一下记录，做到尽可能的完善。当然还有很多意想不到的情况，这就只能在程序运行时，随时发现修改了，这也是一个软件逐渐成熟的过程。

结束语

经过 3 个多月的时间，软件终于完成了，心里稍微放松了一点。但在后面的日子里还会反复琢磨一些处理的过程，有时会突然想到一个情况，就要检查一下代码，看看是否处理逻辑遗漏了什么，这是写代码的状态还在延续。希望我写的这个软件你能读懂，能让你增加知识，增进对 Oracle 数据库的理解，如有疑问，请来邮件一起交流吧。