Datenbanken II B: DBMS-Implementierung

Chapter 8: Storage of Relations I: Segments (Files)

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2021/22

http://www.informatik.uni-halle.de/~brass/dbi21/

Objectives

After completing this chapter, you should be able to:

- write a short paragraph explaining how blocks are allocated in Oracle (mention segments, extents).
- find storage information in the data dictionary.

  And use the `ANALYZE TABLE` command to populate the dictionary tables.
- explain how relations are stored in Oracle (row and block format, TIDs/ROWIDs, migrated rows).
- estimate the number of blocks needed for a table.
- set the basic storage parameters for relations in Oracle for good performance.

Contents

Segments (1)

- If tablespaces are the "logical disks" of Oracle, segments are the "logical files".
- Segments are sequences of data blocks within a tablespace.

  The sequence does not have to be the physical sequence. The blocks are not necessarily stored in contiguous places.

- Segments can grow (blocks can be appended at the end) and shrink (blocks are removed at the end).

  In Oracle, segments shrink only when explicitly requested.

Segments (2)

- The used storage in a tablespace is partioned into segments.

  Every data block can belong to at most one segment.

- A tablespace can contain many segments.
- For every table, Oracle creates a segment inside the tablespace that is mentioned in the `CREATE TABLE`.
- In the same way, each index is stored in a segment.

  The four basic kinds of segments are: Data segments (for tables), index segments, rollback segments (for storing old versions of blocks), temporary segments (for sorting during query evaluation).

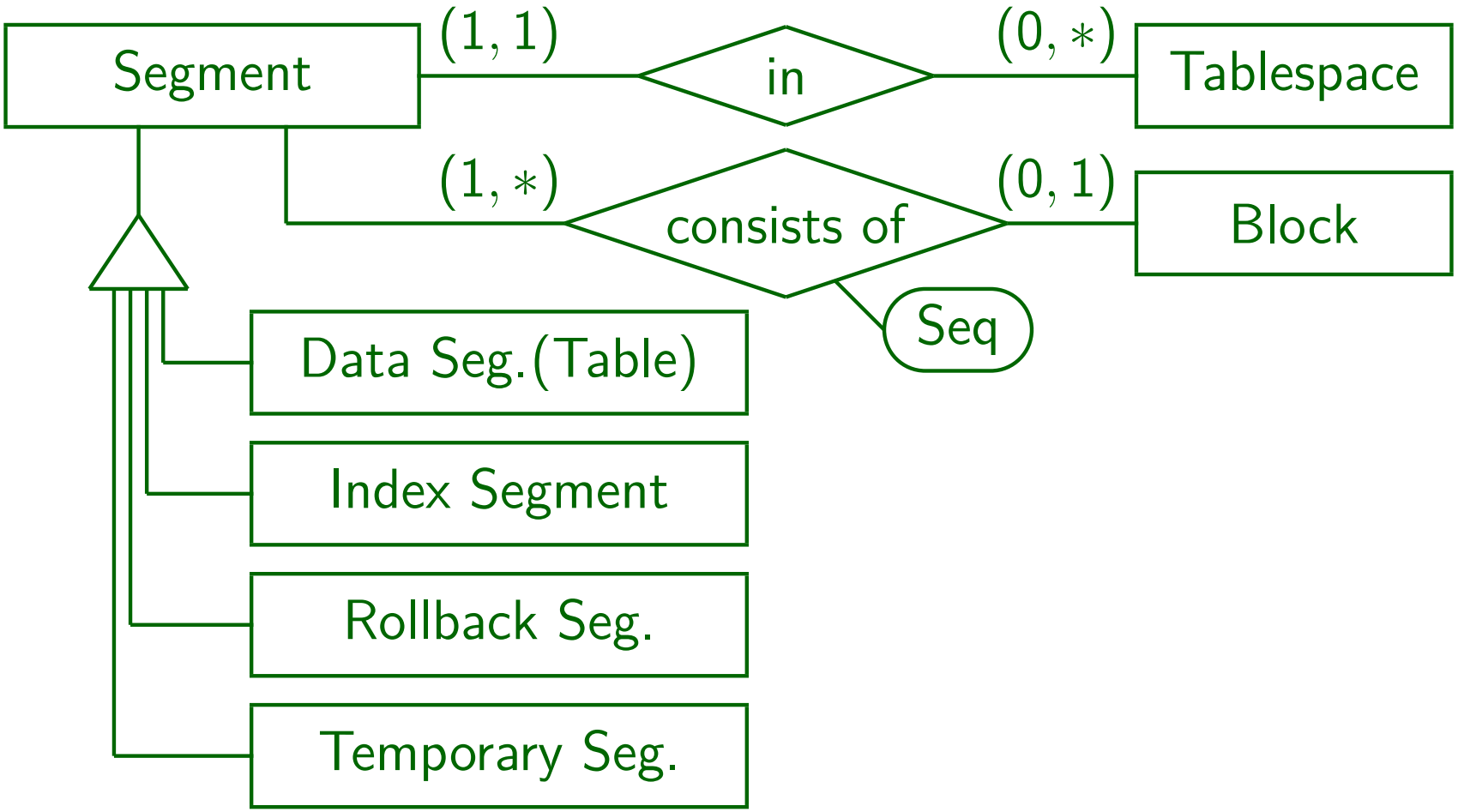
Segments (3)

- Normally, the relationship between data segments and tables is 1:1. But in general, it can be n:m:
  - Partioned tables have more than one segment (usually in different tablespaces on several disks).

    A partitioned table is stored in several pieces, where each piece is basically a table with the same structure: The complete table is then the union of the pieces. When rows are inserted, conditions on the data determine in which piece the row is stored.
  - Clusters can contain rows from several tables having one or more attributes in common.

    Clusters are an Oracle-specific data structure that permits very efficient joins because the rows to be joined together are already stored together (ideally, in the same block).

Segments (4)


Segment
(1, 1)
in
(0, ∗)
Tablespace
(1, ∗)
consists of
(0, 1)
Block
Seq
Data Seg.(Table)
Index Segment
Rollback Seg.
Temporary Seg.

Segments (5)

- The data dictionary view DBA_SEGMENTS contains one row for each segment. It has the following columns:
  - OWNER: User who created the table etc.
  - SEGMENT_NAME: Table name, index name, etc.
  - PARTITION_NAME: For partitioned tables (else null).
  - SEGMENT_TYPE: Type of the segment, e.g. TABLE.

    TABLE, INDEX, CLUSTER, TABLE PARTITION, INDEX PARTITION, ROLLBACK, DEFERRED ROLLBACK, TEMPORARY, CACHE, LOBINDEX, LOBSEGMENT.
  - TABLESPACE_NAME: Tablespace in which the segment is stored.

Segments (6)

- Columns of DBA_SEGMENTS, continued:
  - HEADER_FILE, HEADER_BLOCK: Storage position of segment header block.

    This is the first block of the segment. It contains control information and is not available for table data.
  - BYTES, BLOCKS: Current size of the segment.

    BYTES is simply BLOCKS * DB_BLOCK_SIZE.
  - EXTENTS: Number of storage pieces.
  - INITIAL_EXTENT, NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE: Parameters for allocating storage pieces, see below.

Segments (7)

- Columns of DBA_SEGMENTS, continued:
  - FREELISTS, FREELIST_GROUPS: For management of blocks with free space within the segment.

    Usually both are 1, but if there are many parallel users that insert data, these parameters can be increased.
  - RELATIVE_FNO: File containing seg. header block.

    For Parallel Server (Please explain if you know).
  - BUFFER_POOL: Buffer pool for caching blocks from this segment.
- USER_SEGMENTS lists the segments owned by the current user (some of the above columns are missing).

Extents (1)

- Oracle allocates storage in units called “extents”.
- An extent is sequence of contiguous disk blocks.

  Thus, an extent can be especially fast read from the disk.

- An extent belongs to a single segment and thus to a single table (or index etc.).
- A segment can consist of many extents. But too many extents give bad performance.

  The disk head has to move between the extents (a segment with many extents is “fragmented”). Also, the list of extents should fit into one block. More than 100–500 extents are certainly bad. A single extent would be perfect. One must plan how much space will be needed.

Extents (2)

- Extent sizes are specified in the table declaration:

```
CREATE TABLE STUDENTS(SID NUMERIC(3), ...)
             TABLESPACE USER_DATA
             STORAGE(INITIAL 200K
                     NEXT 50K
                     PCTINCREASE 100)
```

- When the table is created, the initial extent is allocated.

  Although it does not yet contain any rows, it needs disk space for the initial extent (200 KB in the example). The extent size should be a multiple of `DB_BLOCK_SIZE` $*$ `DB_FILE_MULTIBLOCK_READ_COUNT` (the size that Oracle reads during a full table scan in one disk access).

Extents (3)

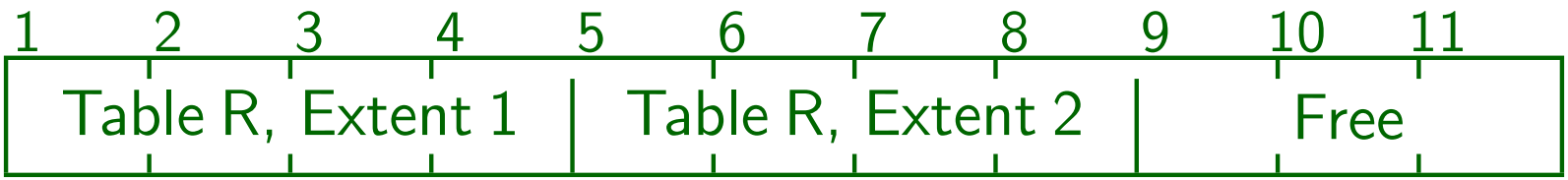
- Whenever the disk space allocated for a table is full, another extent will be allocated.
- In the example, the second extent will be 50 KByte (`NEXT`). Normally, all following extents have this size.
- However, with the parameter `PCTINCREASE` one can request that each following extent will be larger than the previous one (reduces number of extents).

  `PCTINCREASE 100` means that the extent size is doubled. Third extent: 100 KB, fourth: 200 KB, etc. If the extent size grows so fast, there will certainly not be very many extents. However, since one soon gets very large extents, space may be wasted.

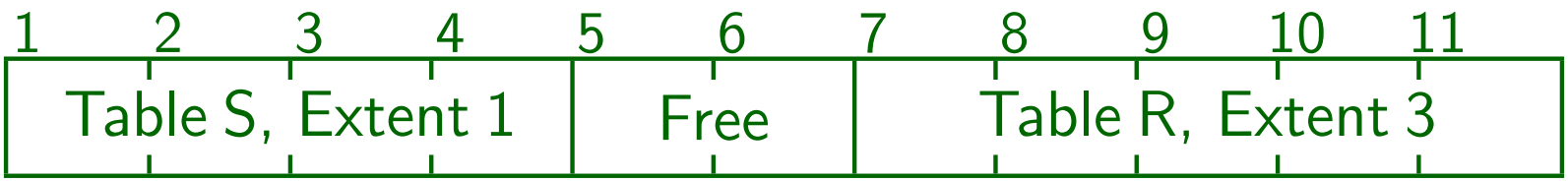
Extents (4)

Example:

File 1:

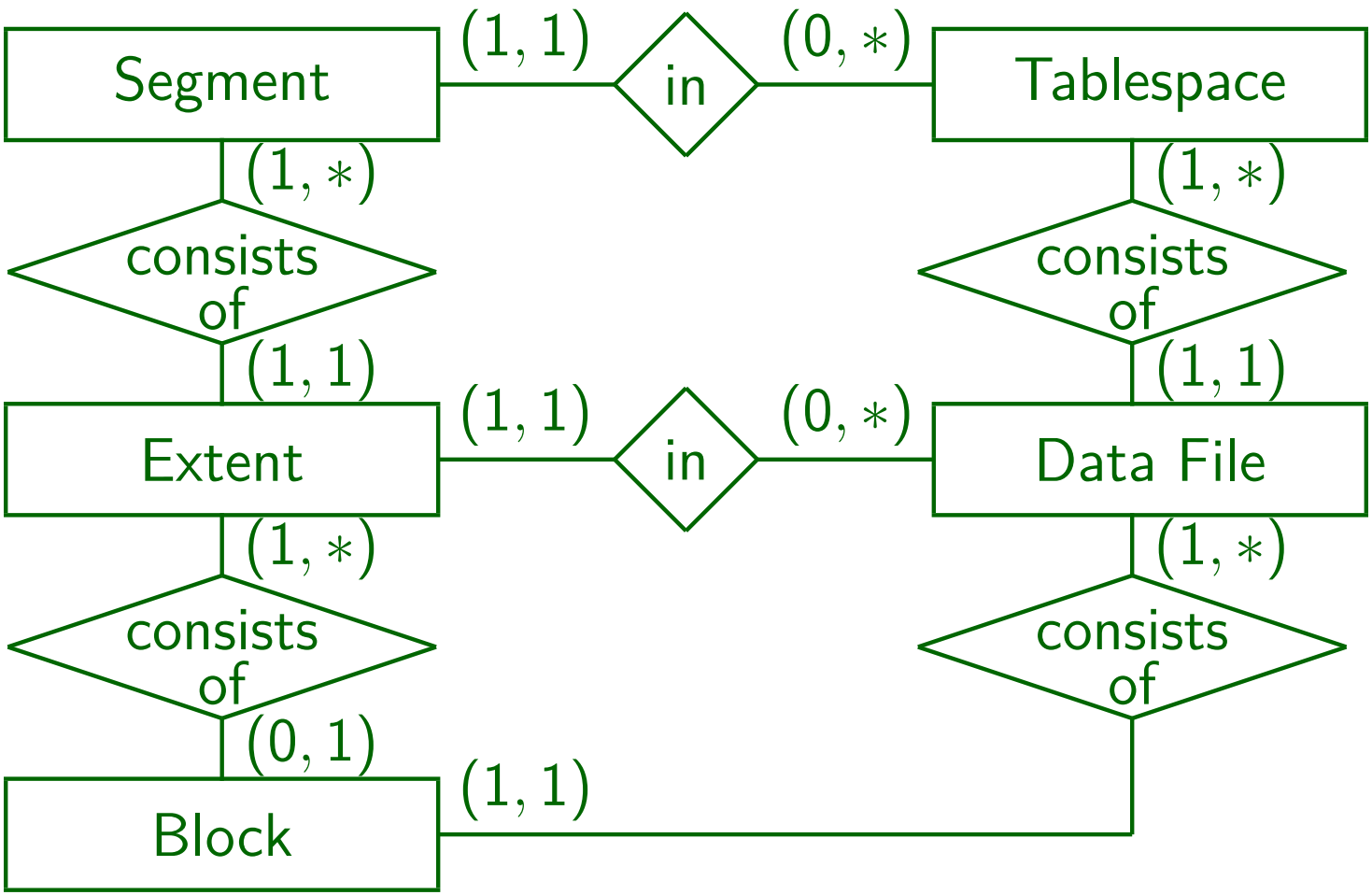


File 2:



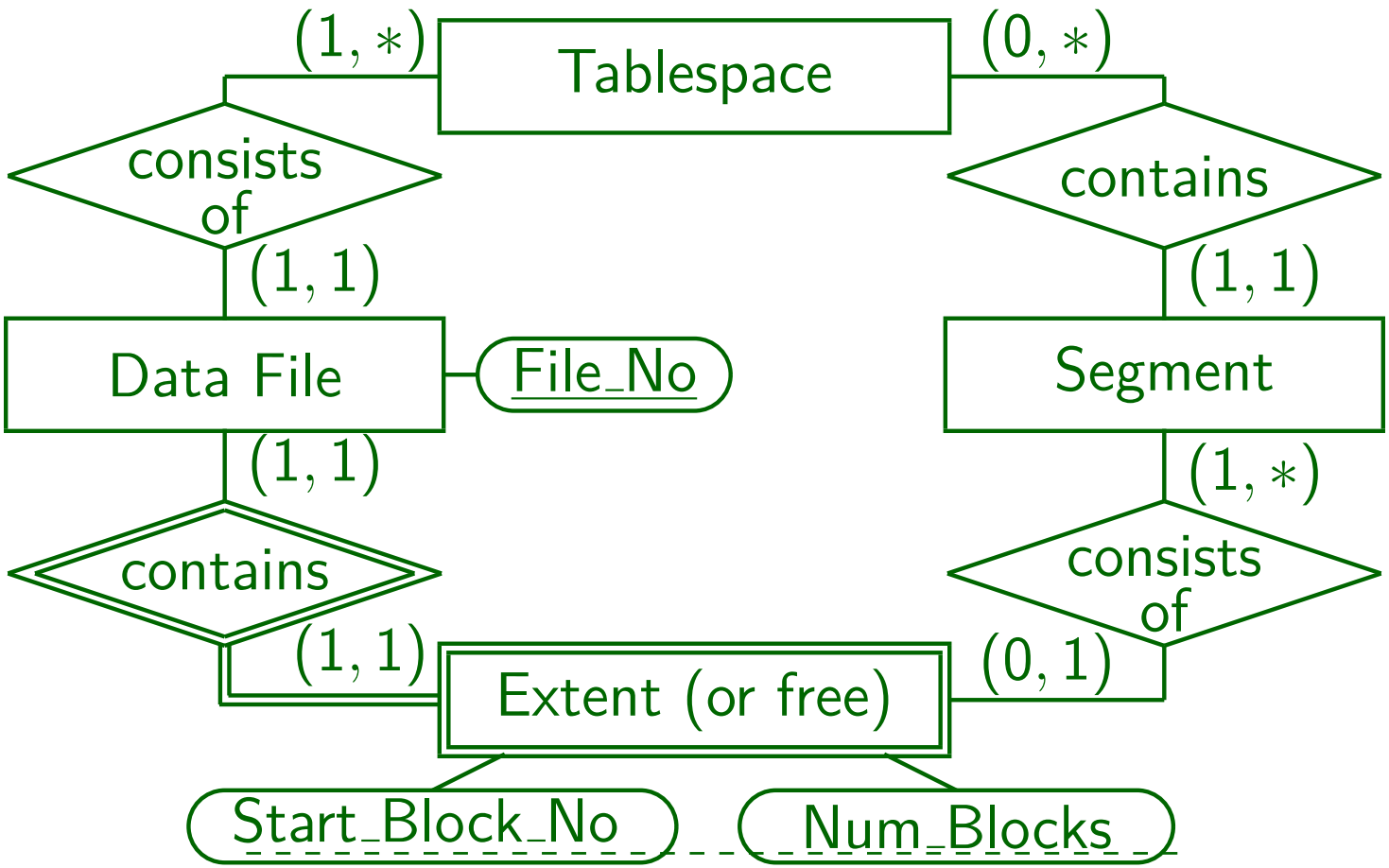
Tables R and S are stored in a tablespace which consists of two data files. Table R has three extents: Block 1 to 4 in File 1, Block 5 to 8 in File 1, and Block 7 to 11 in File 2. Oracle does not merge contiguous extents of a table. Table S consists of a single extent (Block 1 to 4 in File 2).

Extents (5)


Segment
(1, 1)
in
(0, ∗)
Tablespace
(1, ∗)
consists of
(1, ∗)
consists of
(1, 1)
(1, 1)
Extent
(1, 1)
in
(0, ∗)
Data File
(1, ∗)
consists of
(1, ∗)
consists of
(0, 1)
(1, 1)
Block

Extents (6)

Alternative Design:



Extents (7)

- DBA_EXTENTS contains one row for each extent.
  - OWNER, SEGMENT_NAME, PARTITION_NAME: Identification of the segment to which this extent belongs.
  - SEGMENT_TYPE, TABLESPACE_NAME: See DBA_SEGMENTS.
  - EXTENT_ID: Extent number within segment.

    Counted from 0, i.e. $0, 1, 2, \ldots$.
  - FILE_ID: File containing the extent.
  - BLOCK_ID: Start of the extent within the file.
  - BYTES, BLOCKS: Size of the extent.
  - RELATIVE_FNO: Relative file number of first block.

    I am not sure what relative file number means. Please help.

Extents (8)

- DBA_FREE_SPACE contains one row for each contiguous sequence of blocks that is currently not allocated to a segment ("free extents").
  - TABLESPACE_NAME, FILE_ID: Tablespace, data file.
  - BLOCK_ID: First block of free piece.
  - BYTES, BLOCKS: Size of free piece.
  - RELATIVE_FNO: Relative file no of first extent block.

    DBA_FREE_SPACE might contain two adjacent pieces. Oracle checks only from time to time (or when necessary) whether adjacent pieces can be merged ("coalesced").
- See also: USER_EXTENTS, USER_FREE_SPACE.

TS Declaration: Extents (1)

- In the `CREATE TABLESPACE` command, default values for the extent parameters can be specified:

```
CREATE TABLESPACE USER_DATA
DATAFILE 'D:\User1.ora' SIZE 20M
MINIMUM EXTENT 32K
DEFAULT STORAGE (INITIAL 100K NEXT 50K
                 PCTINCREASE 5
                 MINEXTENTS 1 MAXEXTENTS 50
                 BUFFER_POOL KEEP)
```

- `DBA_TABLESPACES` lists these values (used for all segments in the tablespace unless otherwise specified).

TS Declaration: Extents (2)

- The `DEFAULT STORAGE` parameters have no meaning for the tablespace itself, they only apply to tables created within it.
- E.g. if one does not specify `PCTINCREASE` for a table, it will not be 0, but the value defined in the tablespace declaration.

  If one does not define it there, defaults set by Oracle are used: `PCTINCREASE=50`, 5 blocks for `INITIAL` and `NEXT`. The small default values for `INITIAL` and `NEXT` show that at least for large tables, it is important to set these parameters

TS Declaration: Extents (3)

- If one needs to create many similar tables in a tablespace, it is easier to set default values for the tablespace instead of setting the values for each table.
- For temporary segments (created during query evaluation), one cannot explicitly set the physical storage parameters. But default values for the temporary tablespace can be set.

Extent Allocation (1)

- The following is basically the explanation from the Oracle manual.

  Experiments show that extents are sometimes slightly larger than expected.

- First, Oracle searches through the list of all "free extents" of the requested tablespace for an exactly fitting piece of disk space.

  Of course, the requested extent size is rounded up to the next multiple of `DB_BLOCK_SIZE` (or to the minimal extent size declared for the tablespace). The first extent must consist of at least two blocks, because the first block of each segment is the segment header and cannot be used for table data.

Extent Allocation (2)

- If an extent is found, the data dictionary and the segment header are updated to reflect the allocation of the disk space.
- If no free space is found that has a size equal to the requested amount, Oracle searches the list again for a piece that is larger than the requested one.
  - If the first piece found is larger by 5 blocks or more, a piece of the requested size is cut off.
  - If the piece found is larger by less than 5 blocks, it is completely allocated as the new extent.

Extent Allocation (3)

- If all existing pieces of free space are smaller than requested, Oracle merges adjacent pieces. Then both steps are repeated.
- If still no piece is found, and `AUTOEXTEND` is on for at least one data file, the data file is extended (i.e. more disk space is requested from the OS).
- Else the operation fails and an error message is returned ("tablespace full").

Local Extent Management (1)

- Since Oracle 8i, free space can alternatively be managed by an array of bits showing which "extents" are allocated (became default in Oracle 9i).
- For such tablespaces, one can
  - either define a uniform extent size (then one bit is used for each piece of that size)
  - or let Oracle determine the extent size (the algorithm is not disclosed in the documentation).
- The bitmaps are stored in each data file, and not in the data dictionary, thus the name.

Local Extent Management (2)

- Advantages of the new solution with bitmaps:
  - No merging of adjacent pieces of free storage.

    There is only one extent size or a small number of different sizes. If necessary, a sequence of "free"-bits can easily be found.
  - Parallel allocation of extents is possible.

    Before: Possible contention when accessing the free extent list.
  - No undo or redo information is generated.

    The old solution required to change a rollback segment when an extent was reserved.
  - The old method sometimes needed recursive allocation of extents.

    The data dictionary entry might itself need space.

Local Extent Management (3)

- In addition, the user does not have to think about certain parameters when creating tables:
  - The parameters `NEXT` and `PCTINCREASE` are not allowed for tables in such tablespaces.

    They are meaningless because the extent size is now controlled by Oracle. `MINEXTENTS` and `MAXEXTENTS` are excluded, too. The `DEFAULT STORAGE` clause cannot be used for such tablespaces (but if local management is not explicitly specified, `DEFAULT STORAGE` parameters decide between uniform and automatic extent size).
  - `INITIAL` can still be used to define how much space is allocated for the table.

    However, then several extents will be allocated to give the requested total size (the extent size is not influenced).

Local Extent Management (4)

- Creating a tablespace with uniform extent size:

```
CREATE TABLESPACE mydata
DATAFILE '/oradata/mydb/mydata01.dbf' SIZE 100M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;
```

- Alternative (Oracle determines extent sizes):

```
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

  This is now the default setting.

- If one wants the old method:

```
EXTENT MANAGEMENT DICTIONARY
```

This works only in DBs with dictionary-managed `SYSTEM`-tablespace.

Local Extent Management (5)

- The data dictionary table DBA_TABLESPACES has columns (among others):
  - TABLESPACE_NAME: Identifies tablespace.
  - EXTENT_MANAGEMENT: LOCAL or DICTIONARY.
  - ALLOCATION_TYPE: SYSTEM, UNIFORM or USER.

    SYSTEM means AUTOALLOCATE (Oracle determines the extent sizes).
  - SEGMENT_SPACE_MANAGEMENT: MANUAL or AUTO.

    This is for managing blocks with free space within a segment (see below). Do not confuse this with automatic extent management described above.
  - BIGFILE: YES or NO.

Local Extent Management (6)

- The algorithm for `AUTOALLOCATE` is not disclosed and can change at any time.
- One test has shown the following behaviour:
  - First 16 extents: 64 KB (1 MB in total).
  - Next 63 extents: 1 MB (total space is 16 MB).
  - Next 126 extents: 8 MB (until total space 1 GB).
  - Then extent size 64 MB is chosen.

    However, the selected extent size may depend on many factors (e.g. the size of the tablespace, and of existing tables in the tablespace), so the real algorithm is probably more complex. Note also that Oracle seems to think it is ok to have a table with more than 100 extents (then it probably is).

Summary (1)

Tasks of the Disk Manager:

- Create a segment with a given initial size.
- Delete a segment.
- Grow a segment by a given number of blocks.
- Shrink a segment (might not be really required).
- Return all blocks of the segment in the logical sequence (i.e. open scan, read next block, close scan).

Summary (2)

Tasks of the Disk Manager, continued:

- It is also necessary to create pointers to blocks (in indexes):
  - If the segment management never moves blocks, pointers can be physical block addresses.
  - Else needed: "Return *i*-th block of the segment".

    Operating systems usually have a call to set the current position in an open file without reading all intermediate blocks (lseek).
- Auxillary function:
  Free space management for tablespace.

UNIX File System (1)

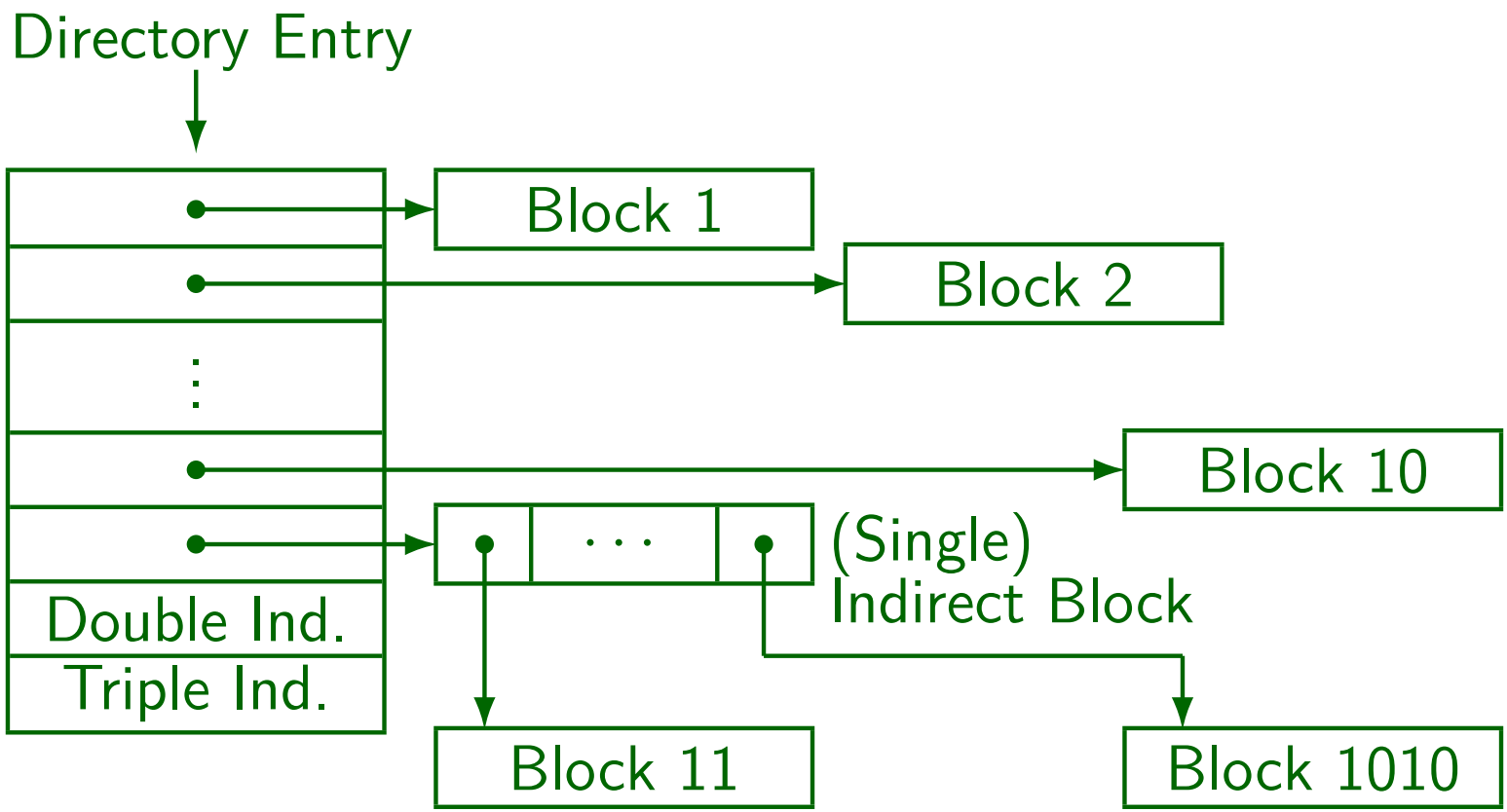
- Segments are very similar to files.
- The file system of any operating system has to implement the same basic functions as the Oracle segment management.
- In UNIX, there are no extents: It manages a list of block addresses for every file.
- It is not a linear list, but a tree (see below).
- In this way, one can efficiently jump to a specific file location.

UNIX File System (2)

- If the blocks of a file were randomly distributed over the entire disk, this would be a big performance problem.
- Therefore, the disk is divided in several cylinder groups (within a cylinder group, the disk head must move only very little).
- If possible, files are kept in a single cylinder group.

  The block allocation routine even allows to specify a block near to which the new block should be allocated.

UNIX File System (3)



References

- Elmasri/Navathe: Fundamentals of Database Systems, 3rd Edition. Section 5.5,5.7.
- Ramakrishnan/Gehrke: Database Management Systems, 2nd Edition. Section 7.3, 7.5–7.8.
- Silberschatz/Korth/Sudarshan: Database System Concepts, 3rd Ed., Chap 10.
- Kemper/Eickler: Datenbanksysteme (in German), Chap. 7, Oldenbourg, 1997.
- Garcia-Molina/Ullman/Widom: Database System Implementation. Chapter 3.
- Härder/Rahm: Datenbanksysteme, Konzepte und Techniken der Implementierung (in German). Chapter 2, 5.
- Jason S. Couchman: Oracle8i Certified Professional: DBA Certification Exam Guide with CDROM. Osborne/ORACLE Press, ISBN 0-07-213060-1, ca. 1257 pages.
- Mark Gurry, Peter Corrigan: Oracle Performance Tuning, 2nd Edition (with disk).
- Gray/Reuter: Transaction Processing: Concepts and Techniques. 1993.
- Oracle 8i Concepts, Release 2 (8.1.6), Oracle Corporation, 1999, Part No. A76965-01.
- Oracle 8i Designing and Tuning for Performance, Release 2 (8.1.6), Oracle Corporation, 1999, Part No. A76992-01.