DBA Certification Course

(Summer 2008)

Chapter 3: Database Components

- Tablespaces
- Buffer Pools
- Schemas
- Catalog

Objectives

After completing this chapter, you should be able to:

- explain the different types of tablespaces
- create a tablespace and a buffer pool
- create a database
- specify tablespaces in a `CREATE TABLE` statement

Literature

- George Baklarz: DB2 9 Fundamentals exam 730 prep, Part 3: Accessing DB2 data

  [http://www.ibm.com/developerworks/edu/dm-dw-db2-cert7303.html]

- Clara Liu, Raul Chong, Dwaine Snow, Sylvia Qi: Understanding DB2: Learning Visually with Examples

  IBM Press/Pearson, 2005, ISBN 0-13-185916-1, 895 pages.

- DB2 for Linux, UNIX, and Windows Version 9 Information Center

  [http://publib.boulder.ibm.com/infocenter/db2luw/v9//index.jsp]

Overview

Storage Concepts (1)

- A tablespace is something like a logical disk, it defines the storage space for tables.
- A tablespace belongs to exactly one database.
- A database can have many tablespaces (at least three: catalog, user, and temporary tablespace).
- A tablespace can contain several tables.
- The main data of a table is stored in exactly one tablespace (???).

    The indexes for a table can be stored in a second tablespace, the large objects (and possibly XML data) in a third.

Storage Concepts (2)

- A tablespace consists of one or more containers.
- A container is a physical location, where data can be stored (a file, directory, or disk).
- A container belongs to exactly one tablespace.
- A container consists of extents, which in turn consist of pages.
- A page is the smallest unit that can be read from or written to the disk.
- An extent is a consecutive sequence of pages.

Storage Concepts (3)

- A buffer pool is an area in main memory used for caching pages.

  Of course, it should be in real memory, not virtual memory. Otherwise double paging occurs and the DBMS looses the control on which pages are currently in memory.

- A database can have several buffer pools.
- Each tablespace can be assigned to only one buffer pool. Several tablespaces can be assigned to the same buffer pool.
- Exercise: Draw an ER-diagram for the concepts: Database, tablespace, container, table, buffer pool.

Default DB Components

- When a database is created, by default three table spaces are created:
  - ◇ `SYSCATSPACE`: Contains system catalog tables.
  - ◇ `TEMPSPACE1`: Temporary storage location for large sort or join operations.
  - ◇ `USERSPACE1`: Default location for user tables.

    If there are several tablespaces available for user tables, quite complicated rules are used for selecting a table space when the `CRATE TABLE` statement does not explicitly specify one. See below.
- In addition, one buffer pool is created:
  - ◇ `IBMDEFAULTBP`

CREATE TABLE

- Tablespaces can be defined for a table as follows:

```
CREATE TABLE mytable (
    mycol1 numeric(2) not null primary key,
    ...
)
IN mytbspace1
INDEXES IN mytbspace2
LONG IN mytbspace3
```

- One must have the `USE` right for the tablespaces.

  This is the only possible right for a tablespace object. The grants for tablespaces are listed in `SYSCAT.TBSPACEAUTH`. The tablespaces for a table are listed in `SYSCAT.TABLES` (columns `TBSPACE`, `INDEX_TBSPACE`, `LONG_TBSPACE`).

Page Size (1)

- DB2 supports four page sizes: `4K`, `8K`, `16K`, `32K`. The default is `4K`.
- The page size is important because each row must fit in a single page (with the exception of LOB columns and possibly XML columns).

  With 4K page size, the maximum size of a row including all overhead is 4005 Byte. It can have at most 500 columns. With 8K page size, the maximum size of a row is 8101 Byte, and it can have 1012 rows.

- The page size is also important because there is a maximal number of pages in a tablespace.

  So very large tables might need a larger pagesize. See Slide 3-29.

Page Size (2)

- For regular table spaces (see Slide 3-29), there can be only 255 rows per page.

  Thus, a page size that is too large means that storage is wasted.

- Each tablespace has a fixed page size.
- A database can have tablespaces with different page sizes.
- Each buffer pool has a fixed page size.
- A tablespace can be assigned only to a buffer pool with the same page size.

Extents

- An extent is a sequence of consecutive pages.

  Sequential block access is much faster than random access.

- All extents in a table space have the same size.
- DB2 allocates storage in units of extents.

  For SMS tablespaces, this depends on `multipage_alloc`, see below.

- Each extent belongs to only one database object (e.g., one table).
- If a table space has several containers, extents are allocated in a round robin fashion.

  This gives striping if the containers are on different disks.

Overview

Tablespace Types (1)

- Tablespaces can be classified according to the way storage is managed:
  - ◇ SMS ("System Managed Space"): Data is stored in single files (e.g., one per table).

    One specifies one or more directories for an SMS tablespace. The operating system manages the blocks. SMS was default in Ver. 8.
  - ◇ DMS ("Database Managed Space"): Data is stored in one or a few large files (or raw devices).

    The DBMS manages the assignment of storage blocks to tables.
  - ◇ Automatic Storage: DBMS decides on files.

    One specifies only one or more storage paths to be used for all tablespaces with automatic storage.

Tablespace Types (2)

- Furthermore, tablespaces can be classified according to their contents into
  - ◇ Regular Tablespaces:
    For tables, indexes, etc. (not too big).
  - ◇ Large Tablespaces:
    For tables, indexes, etc. (big).
  - ◇ System Temporary Tablespaces:
    For sorts, joins.
  - ◇ User Temporary Tablespaces:
    For temporary tables.

SMS Tablespaces (1)

- An SMS tablespace can be created e.g. with the command

```
CREATE TABLESPACE myspace MANAGED BY SYSTEM
                          USING ('C:\my_dir')
```

- When an object (e.g., a table) is created, it is assigned an object-ID.
- Suppose a table has object-ID `00001`, then its
  - ◇ data are stored in `C:\my_dir\SQL00001.DAT`,
  - ◇ indexes are stored in `C:\my_dir\SQL00001.IDX`.

    Other extensions: `.BKM` for block indexes, `.LF` for `LONG VARCHAR` columns, `.LB` for LOB data, and `.LBA` for addresses in the `.LB`-file.

SMS Tablespaces (2)

- One can specify also a relative path:

```
CREATE TABLESPACE myspace MANAGED BY SYSTEM
                          USING ('myspace_dir')
```

- Under Windows, the directory will be, e.g.

```
C:\DB2\NODE0000\SQL00001\myspace_dir
```

  ◇ `C:` is the drive specified in `create database`,

  ◇ `DB2` is the name of the instance,

  ◇ `NODE0000` is important only for partitioned DBs,

  ◇ and `SQL00001` is the directory for the first DB.

    UNIX: `/db/db2inst1/NODE0000/SQL00001/myspace_dir`, where `/db` is given in the `create database` and `db2inst1` is the default instance.

SMS Tablespaces (3)

- DB2 automatically creates the directories.
- It is possible to specify several containers:

```
CREATE TABLESPACE myspace MANAGED BY SYSTEM
                          USING ('dir1', 'dir2')
```

- Then all files are created in both directories and striping is done.
- One cannot add a container later to an SMS table-space (except possibly in a partitioned DB).

  Therefore, it is important to choose a directory on a device that is large enough. Multiple containers should have the same size.

SMS Tablespaces (4)

- SMS Tablespaces are easy to administer.

  The are not many options to choose, the files grow and shrink on demand.

- SMS tablespaces are less performant than DMS tablespaces.

- Some possibilities that exist for DMS tablespaces to not exist for SMS tablespaces.

  E.g. storing indexes and large objects separately (in a different tablespace, probably on a different disk).

- SMS tablespaces are limited to 64 GB if 4K pages are used (there are no large SMS tablespaces).

SMS Tablespaces (5)

- For SMS tablespaces, it might not be true that DB2 always allocates complete extents:
  - ◇ If the DB CFG parameter `multipage_alloc` is set to `NO`, the files for the tables are extended only one page at a time.

    Then the efficiency depends on good defragmentation mechanisms in the operating system.
  - ◇ Starting from Version 8.2, this parameter is set to `YES` by default, which means that always a complete extent is added to the files.

DMS Tablespaces (1)

- A DMS tablespace can be created with:

```
CREATE TABLESPACE myspace MANAGED BY DATABASE
       USING (FILE 'C:\my_file' 1000)
```

- The file will be 1000 pages large.

  E.g. 4M for 4K pages. It is also possible to specify the size directly in K, M, G. Then e.g. 1M means 1 megabyte, not one million pages. Note that one extent is used for the container tag (see below).

- If the file does not exist yet, it is created with the given size.

  If it exists, DB2 tries to make sure that it is currently unused, then it is made the correct size.

DMS Tablespaces (2)

- One can specify several files, then striping is done:

```
CREATE TABLESPACE myspace MANAGED BY DATABASE
       USING (FILE 'C:\my_file1' 1000,
              FILE 'D:\my_file2' 1000)
       EXTENTSIZE 32
```

- Instead of "`FILE`", one can also use "`DEVICE`".

  For a disk or partition. Raw devices might give about 10–15% better performance (at the expense of more difficult administration).

- If the file size is not a multiple of the extent size, the remaining pages are not used.

  Tablespaces must be at least 5 extents large (+1 for container tag).

DMS Tablespaces (3)

- It is recommended that all files for a DMS table-space are of equal size.

  If they are not, the files are aligned at the start, and size of a stripe (number of containers) becomes smaller when a file ends.

- It is possible to add a container to a DMS table-space. Then "rebalancing" is done, i.e. extents are moved to use the new container for striping.

```
ALTER TABLESPACE myspace
      ADD (FILE 'E:\my_file3' 1000)
```

  If the new file should be smaller than the existing ones, its end is aligned with the end of the largest file. To avoid rebalancing (adds file after all existing): `ALTER TABLESPACE ... BEGIN NEW STRIPE SET (...)`

DMS Tablespaces (4)

- One can also change the size of an existing file:

```
ALTER TABLESPACE myspace
      EXTEND (FILE 'C:\my_file' 500)
```

If the file had 1000 pages large before, it will now have 1500 pages.

- One can also explicitly specify the new size:

```
ALTER TABLESPACE myspace
      RESIZE (FILE 'C:\my_file' 1500)
```

- One can free pages that were never used (after the “high water mark”):

```
ALTER TABLESPACE myspace
      REDUCE (FILE 'C:\my_file' 500)
```

Automatic Storage (1)

- For automatic storage, paths or devices are specified for the database as a whole, and DB2 itself manages containers for the tablespaces.

  Automatic storage must be enabled for the database if tablespaces in it are to be managed by automatic storage. This can be done only at database creation.

- Automatic storage was added in Version 8.2.2 and became the default in Version 9.

  It is a general trend that DBMS need less administrative work. The DBMS might automatically reconfigure the number and location of containers for an automatic storage tablespace.

Automatic Storage (2)

- Example database creation with automatic storage:

```
CREATE DATABASE mydb
    AUTOMATIC STORAGE YES
    ON /data1, /data2
    DBPATH ON /mydb
    USING CODESET UTF-8 TERRITORY US
    PAGESIZE 8K
    DFT_EXTENT_SIZE 8
```

- The two storage paths `/data1` and `/data2` are used for tablespaces with automatic storage.
- The path `/mydb` is used for various control files.

  If one does not specify a DB path, the first storage path is used.

Automatic Storage (3)

- "`AUTOMATIC STORAGE YES`" is anyway the default.

  Only if it should be disabled, one must specify `AUTOMATIC STORAGE NO`. Note that it cannot be enabled later.

- Even when automatic storage is enabled, one can create tablespaces that are not managed by automatic storage.

- The shortest database creation command is:

```
CREATE DATABASE mydb
```

  In this case the DBM CFG parameter `dftdbpath` determines the database and storage path.

Automatic Storage (4)

- One cannot specify containers when creating a tablespace managed by automatic storage, but one can specify the size:

```
CREATE TABLESPACE myspace
    MANAGED BY AUTOMATIC STORAGE
    AUTORESIZE YES
    INITIALSIZE 100M
    INCREASESIZE 50M
    MAXSIZE NONE
```

Regular/large tablespaces are generated as DMS tablespaces, and temporary tablespaces are generated as SMS tablespaces. This might change in a future release.

Regular vs. Large TBS (1)

- For a regular tablespace with 4K pages, the size limit is 64 GB.

  It grows with the page size, e.g. 128 GB for 8K pages, and so on. I.e. the limit is 16 M pages per regular tablespace (24 bits).

- In a regular tablespace, only 255 rows per page can be stored.

- For a large tablespace with 4K pages, the size limit is 2 TB.

  The limit is now 512 M pages. With 32K pages, this gives 16 TB.

Regular vs. Large TBS (2)

- There is a price to pay:
  - ◇ Indexes need two additional bytes per entry in a large tablespace.
  - ◇ Dropped table recovery is supported only for regular tablespaces.
- SMS tablespaces can only be regular.
- For DMS tablespaces, one can choose:
  - ◇ `CREATE REGULAR TABLESPACE myspace ...`
  - ◇ `CREATE LARGE TABLESPACE myspace ...`

  The default for DMS tablespaces is `LARGE`, for SMS `REGULAR`.

Temporary Tablespaces (1)

- Besides regular and large tablespaces, there are two more types of tablespaces:
  - ◇ `SYSTEM TEMPORARY` tablespaces store temporary data during query evaluation (for sorts and joins)

    The keyword `TEMPORARY` alone is sufficient, `SYSTEM` is the default.
  - ◇ `USER TEMPORARY` tablespaces store declared global temporary tables.
- Such tablespaces can be created with:
  - ◇ `CREATE [SYSTEM] TEMPORARY TABLESPACE mytmp ...`
  - ◇ `CREATE USER TEMPORARY TABLESPACE mytmp ...`

Temporary Tablespaces (2)

- A system temporary tablespace `TEMPSPACE1` is created by default when the database is created.

  If additional system temporary are created they are used in round robin fashion. A database must always have at least one system temporary tablespace.

- Declared global temporary tables can only be used when a database has a user temporary tablespace.

  Since a database has none by default, one must explicitly create such a tablespace if one wants to use declared global temporary tables.

- Temporary tablespaces can be SMS or DMS tablespaces, and can use automatic storage.

Prefetching

- DB2 tries to get pages into the buffer pool that are very likely needed in the near future.
- E.g., for a sequential table scan, it is clear that the pages that follow the currently requested page will also be needed.
- When multiple row identifiers (RIDs) are returned from an index scan, all pages that contain these rows will be needed.

  DB2 sorts the RIDs (by page number), and then fetches the pages.

Tablespace Parameters (1)

- `PAGESIZE`: pagesize (4K, 8K, 16K, 32K).

  The default is/was 4K. Starting in Version 9.1, the default can be set in the `CREATE DATABASE` command. If one wants something else than the default, one must first create a bufferpool with that pagesize.

- `EXTENTSIZE`: extent size in pages.

  One can define the size also in K or M, then it is rounded down to the next multiple of the page size. Default `DFT_EXTENT_SZ` (in DB CFG).

- `PREFETCHSIZE`: how many pages are fetched at once.

  With "`PREFETCHSIZE AUTOMATIC`" the system chooses the number of containers times the extent size. (actually times the number of physical devices per container, set in the registry variable `DB2_PARALLEL_IO`, but usually this is 1). Default: `DFT_PREFETCH_SZ` (in DB CFG).

Tablespace Parameters (2)

- `OVERHEAD`: Seek and latency time in milliseconds.

  For optimizer estimates. Default (in Version 9): 7.5

- `TRANSFERRATE`: Time for one block in milliseconds.

  For optimizer estimates. Default (in Version 9): 0.06

- `NO FILESYSTEM CACHING`: I/O bypasses the OS cache.

  The default is `FILESYSTEM CACHING`, i.e. the OS cache is used.

- `DROPPED TABLE RECOVERY`: Whether dropped tables can be restored during recovery.

  `ON` or `OFF`. Since Version 8, the default is `ON`.

Tablespace Parameters (3)

- `AUTORESIZE`: DBMS may expand containers.

  Default is `YES` for automatic storage tablespaces, `NO` for DMS tablespaces. For SMS tablespaces, the size parameters cannot be defined (all available space on the device is used).

- `INITIALSIZE`: Initially allocated disk space.

  This can be specified only for automatic storage tablespaces.

- `INCREASESIZE`: Size increment if tablespace is full.

  It can be specified as absolute value or as percent of the current tablespace size. It can be specified only if `AUTORESIZE` is on (`YES`).

- `MAXSIZE`: Limit for automatic resizing.

  An absolute value or the keyword `NONE` can be specified.

Tablespace Parameters (4)

```
CREATE REGULAR TABLESPACE myspace
    PAGESIZE 8K
    MANAGED BY DATABASE USING (FILE '/data1' 1M)
    AUTORESIZE YES
    INCREASESIZE 50 PERCENT
    MAXSIZE NONE
    EXTENTSIZE 16
    PREFETCHSIZE 16
    BUFFERPOOL MYBUFFERPOOL
    OVERHEAD 5.2
    NO FILE SYSTEM CACHING
    TRANSFERRATE 0.2
    DROPPED TABLE RECOCERY ON
```

Settings for Standard TBS

```
CREATE DATABASE mydb
    DFT_EXTENT_SZ 16
    CATALOG TABLESPACE MANAGED BY DATABASE
        USING (FILE '/data/catalog' 4000)
        EXTENTSIZE 8
    TEMPORARY TABLESPACE MANAGED BY SYSTEM
        USING ('/data/tmp')
    USER TABLESPACE MANAGED BY DATABASE
        USING (FILE '/data/user' 10M)
        AUTORESIZE YES
        INCREASESIZE 5M
```

Info about Tablespaces

- `LIST TABLESPACES [SHOW DETAIL]`
- `GET SNAPSHOT FOR TABLESPACES ON ⟨database⟩`
- Tablespace containers:

  ```
  db2 list tablespace containers for ⟨TBspace-ID⟩
  ```

  TBSpace-ID is the numerical code of the tablespace (listed with `LIST TABLESPACES`), e.g. 0.

- `SELECT * FROM SYSCAT.TABLESPACES`

  Columns: `TBSPACE`, `OWNER`, `CREATE_TIME`, `TBSPACEID`, `TBSPACETYPE`, `DATATYPE`, `EXTENTSIZE`, `PREFETCHSIZE`, `OVERHEAD`, `TRANSFERRATE`, `PAGESIZE`, `DBPGNAME`, `BUFFERPOOLID`, `DROP_RECOVERY`, `NGNAME`, `DEFINER`, `REMARKS`.

Overview

Container Tags

- DB2 stores some information about the tablespace in each container. This is the "container tag":
    - ◇ In an SMS tablespace, it is stored in the file `SQLTAG.NAM` (in each container/directory).
    - ◇ In a DMS tablespace, it uses the first extent of each container.

        In Version 7, it used only a single page (and one can still get this behaviour with "`db2set DB2_USE_PAGE_CONTAINER_TAG=ON`"). However, when the files are stored on a RAID system, it might be important that the extents in the file are aligned with the stripes of the RAID system, so that each extent is stored completely on one disk. Therefore, from Version 8, an entire extent is reserved for the container tag.

DMS Tablespace Maps (1)

- The usable pages in a tablespace are numbered from 0 to $n-1$ (where n is the number of usable pages). This is the logical page number.
- For each table, (at least) one extent is reserved that contains the extent map:
  - ◇ This contains the positions (logical page number) of all extents of the table.
  - ◇ The last page contains indirect references (to extents that continue the extent map) and double indirect references in the last 16 entries.

DMS Tablespace Maps (2)

- Each tablespace contains an object table:
  - ◇ This is an internal relational table that maps object identifiers (of the tables etc. stored in the tablespace) to the position of the extent map for that table etc.
  - ◇ In order to find the object table, the second page of the tablespace contains the extent map for the object table.

    The first page contains a tablespace header. I assume that these are logical page numbers, which start only after the container tag.

DMS Tablespace Maps (3)

- Logical page numbers in the tablespace are mapped to physical pages in a specific container by means of the tablespace map.
- This is necessary, because containers may be of different sizes, and added later to a tablespace.
- One can view the tablespace map with `GET SNAPSHOT FOR TABLESPACES ON sample`.

  It shows a number of ranges, one per line. For each range, the maximal (logical) extent number in that range is shown (the first range starts with 0, other extents start with the maximal extent number in the previous range +1). For each range, the containers used are listed.

DMS Tablespace Maps (4)

- So at minimum, a tablespace with one table `T1` needs five extents:

  0. Tablespace header and object table extent map.

     Tablespace header in the first page, object table extent map in the second page (refers to Extent 2), rest unused.

  1. Tablespace map (SMPs: space map pages).
  2. Object table (maps `T1` to Extent 3).
  3. Extent map of `T1` (single entry: Extent 4).
  4. Data of the table `T1`.

- Plus one extent is needed for the container tag.

Example (1)

- Tablespace creation:

```
CREATE TABLESPACE sbts
  MANAGED BY DATABASE USING (FILE '/tmp/sbts' 24)
  EXTENTSIZE 4
```

It was not possible to create a tablespace of only 20 pages.

- The output of

```
GET SNAPSHOT FOR TABLESPACES ON sample
```

is shown on the next slides.

- In this tablespace, one can create a table, but only without a key (index): That needs at least 32 pages.

Example (2)

```
Tablespace name                       = SBTS
  Tablespace ID                       = 7
  Tablespace Type                     = Database managed space
  Tablespace Content Type             = All permanent data.
                                        Large table space.
  Tablespace Page size (bytes)        = 4096
  Tablespace Extent size (pages)      = 4
  Automatic Prefetch size enabled     = Yes
  Buffer pool ID currently in use     = 1
  Buffer pool ID next startup         = 1
  Using automatic storage             = No
  Auto-resize enabled                 = No
  File system caching                 = Yes
  Tablespace State                    = 0x'00000000'
   Detailed explanation:
     Normal
```

Example (3)

Empty Tablespace:

```
Tablespace Prefetch size (pages)  = 4
Total number of pages             = 24
Number of usable pages            = 20
Number of used pages              = 12
Number of pending free pages      = 0
Number of free pages              = 8
High water mark (pages)           = 12
Current tablespace size (bytes)   = 98304
Rebalancer Mode                   = No Rebalancing
Minimum Recovery Time             =
Number of quiescers               = 0
Number of containers              = 1
```

Example (4)

After creating a table (without index):

```
Tablespace Prefetch size (pages)   = 4
Total number of pages              = 24
Number of usable pages             = 20
Number of used pages               = 20
Number of pending free pages       = 0
Number of free pages               = 0
High water mark (pages)            = 20
Current tablespace size (bytes)    = 98304
Rebalancer Mode                    = No Rebalancing
Minimum Recovery Time              =
Number of quiescers                = 0
Number of containers               = 1
```

Example (5)

```
Container Name                      = /tmp/sbts
    Container ID                    = 0
    Container Type                  = File (extent sized tag)
    Total Pages in Container        = 24
    Usable Pages in Container       = 20
    Stripe Set                      = 0
    Container is accessible         = Yes

Table space map:

 Range  Stripe Stripe Max    Max  Start  End    Adj. Containers
 Number Set    Offset Extent Page Stripe Stripe
 [   0] [   0]      0      4   19      0      4    0   1 (0)
```

Multiple Containers (1)

- Tablespace with two containers:

```
CREATE TABLESPACE SBTS
   MANAGED BY DATABASE USING
   (FILE '/tmp/sbts1' 34, FILE '/tmp/sbts2' 50)
   EXTENTSIZE 4
```

- `'/tmp/sbts1'` is container 0, usable pages: 28
- `'/tmp/sbts2'` is container 1, usable pages: 44
- Tablespace map:

```
Range  Stripe Stripe Max    Max  Start  End    Adj Containers
Number Set    Offset Extent Page Stripe Stripe
[   0] [   0]      0     13   55      0      6   0    2 (0,1)
[   1] [   0]      0     17   71      7     10   0    1 (1)
```

Multiple Containers (2)

- The first range uses both containers (all 28 usable pages from container 0, and the first 28 usable pages from container 1).
- The first range consists of 7 stripes (number 0 to 6). Each of these consists of two extents (one in each container). Thus, the range has 56 pages.
- The second range has 4 stripes (number 7 to 10) of one extent each (using only container 1). Thus, this range has 16 pages (pages 56 to 71).

Multiple Containers (3)

- Now a third container is added as a new stripe set:

```
ALTER TABLESPACE SBTS
  BEGIN NEW STRIPE SET (FILE '/tmp/sbts3' 24)
```

- Tablespace map:

```
Range   Stripe Stripe Max    Max  Start  End    Adj Containers
Number  Set    Offset Extent Page Stripe Stripe
[   0]  [   0]      0     13   55      0      6  0    2 (0,1)
[   1]  [   0]      0     17   71      7     10  0    1 (1)
[   2]  [   1]     11     22   91     11     15  0    1 (2)
```

- This container (ID 2) has 20 usable pages, which are added as stripe 11 to 15 (one extent each).