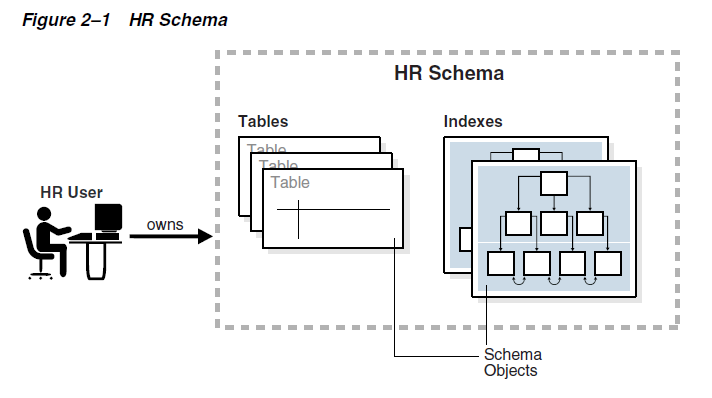
**Oracle database concepts**

A database **schema** is a logical container for data structures, called **schema objects**. Examples of schema objects are tables and indexes



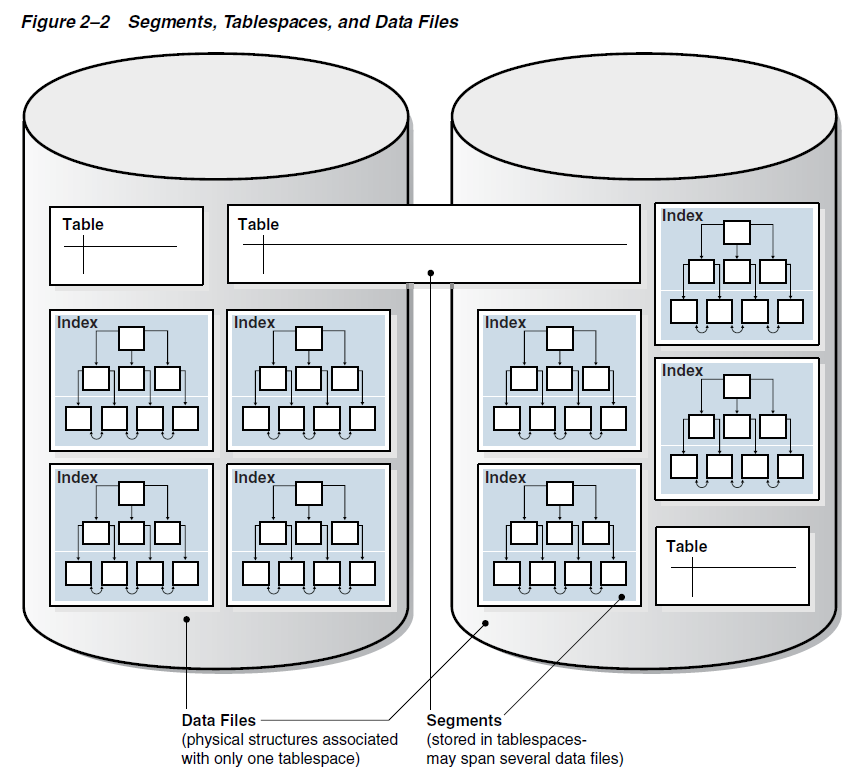
The most important schema objects in a relational database are tables. A **table** stores data in rows.

Oracle SQL enables you to create and manipulate many other types of schema objects, including the following: Indexes, Views, Sequences, Synonyms, PL/SQL subprograms etc.

Some schema objects store data in logical storage structures called **segments**. For example, a nonpartitioned heap-organized table or an index creates a segment. Other schema objects, such as views and sequences, consist of **metadata only**.

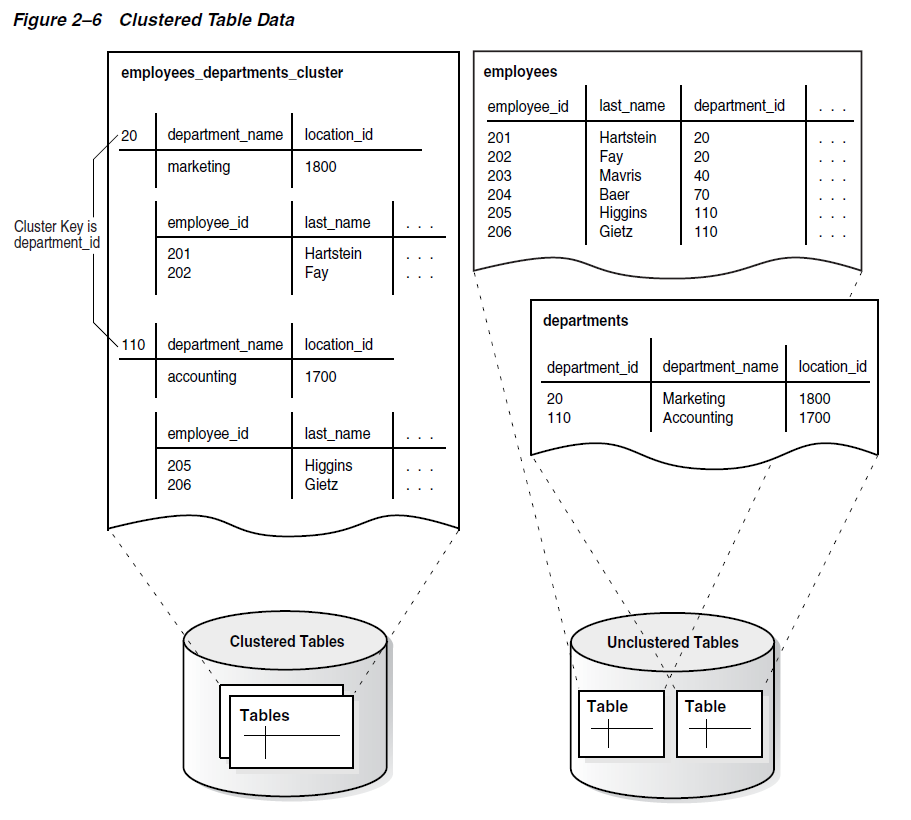
At the operating system level, Oracle Database stores database data in **data files**. For ease of administration, Oracle Database allocates space for user data in **tablespaces**, which like **segments** are logical storage structures. Each segment belongs to only one tablespace. For example, the data for a nonpartitioned table is stored in a single segment, which is turn is stored in one tablespace.

A permanent tablespace contains persistent schema objects. Objects in permanent tablespaces are stored in data files. A temporary tablespace contains schema objects only for the duration of a session.



A **table cluster** is a group of tables that share common columns and store related data in the same blocks. When tables are clustered, a single data block can contain rows from multiple tables. For example, a block can store rows from both the employees and departments tables rather than from only a single table.

The cluster key is the column or columns that the clustered tables have in common. For example, the employees and departments tables share the department\_id column. You specify the cluster key when creating the table cluster and when creating every table added to the table cluster.

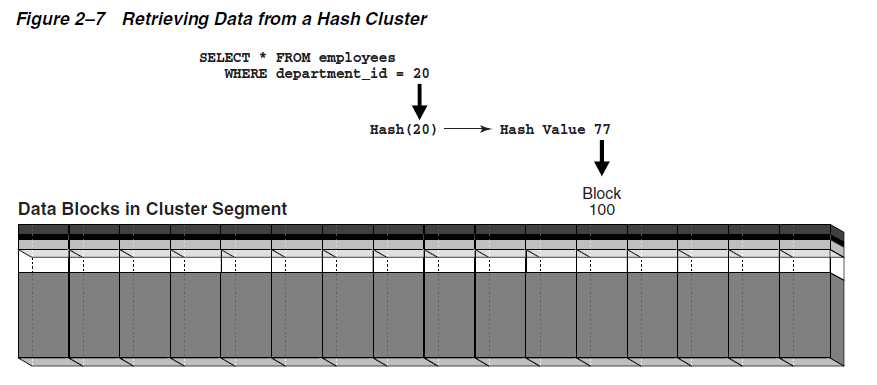


A **hash cluster** is like an indexed cluster, except the index key is replaced with a hash function. No separate cluster index exists. In a hash cluster, the data is the index.

With an indexed table or indexed cluster, Oracle Database locates table rows using key values stored in a separate index. To find or store a row in an indexed table or table cluster, the database must perform at least two I/Os:

* One or more I/Os to find or store the key value in the index
* Another I/O to read or write the row in the table or table cluster

To find or store a row in a hash cluster, Oracle Database applies the hash function to the cluster key value of the row. The resulting hash value corresponds to a data block in the cluster, which the database reads or writes on behalf of the issued statement.

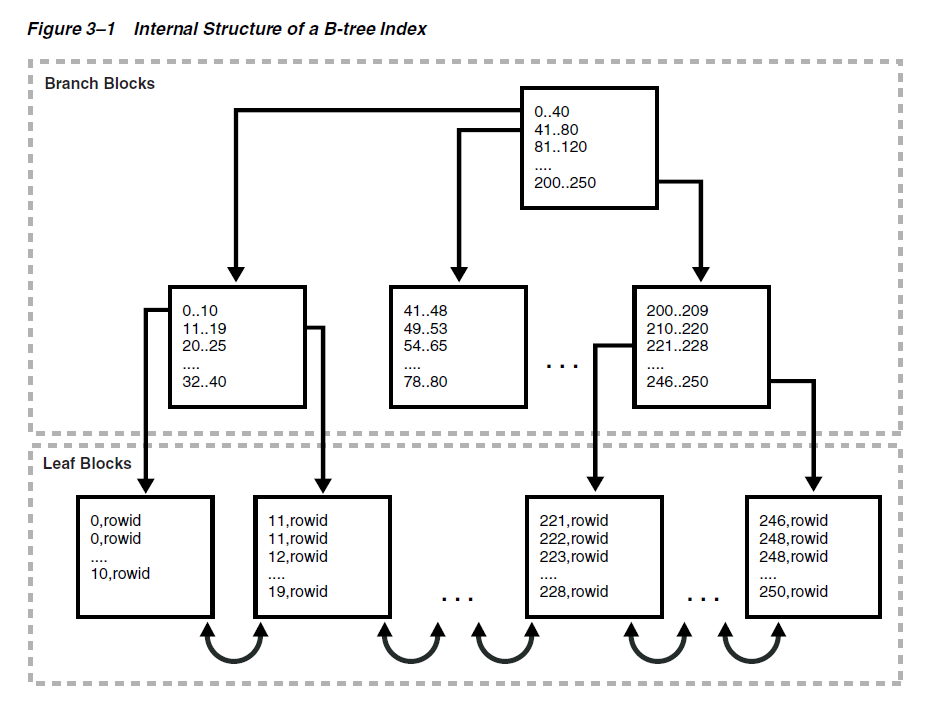


An **index** is an optional structure, associated with a table or table cluster, that can sometimes speed data access. By creating an index on one or more columns of a table, you gain the ability in some cases to retrieve a small set of randomly distributed rows from the table. Indexes are one of many means of reducing disk I/O.

A key is a set of columns or expressions on which you can build an index. Although the terms are often used interchangeably, indexes and keys are different. Indexes are structures stored in the database that users manage using SQL statements. Keys are strictly a logical concept.

A composite index, also called a concatenated index, is an index on multiple columns in a table.

A B-tree index has two types of blocks: branch blocks for searching and leaf blocks that store values. The upper-level branch blocks of a B-tree index contain index data that points to lower-level index blocks.



**Index Scan**

In an index scan, the database retrieves a row by traversing the index, using the indexed column values specified by the statement. If the database scans the index for a value, then it will find this value in n I/Os where n is the height of the B-tree index. This is the basic principle behind Oracle Database indexes.

**Full Index Scan**

In a full index scan, the database reads the entire index in order.

**Fast Full Index Scan**

A fast full index scan is a full index scan in which the database accesses the data in the index itself without accessing the table, and the database reads the index blocks in no particular order.

**Index Range Scan**

An index range scan is an ordered scan of an index that has the following characteristics:

One or more leading columns of an index are specified in conditions.

**Index Unique Scan**

In contrast to an index range scan, an index unique scan must have either 0 or 1 rowid associated with an index key. The database performs a unique scan when a predicate references all of the columns in a UNIQUE index key using an equality operator. An index unique scan stops processing as soon as it finds the first record because no second record is possible.

**Index Skip Scan**

An index skip scan uses logical subindexes of a composite index. The database "skips" through a single index as if it were searching separate indexes. Skip scanning is beneficial if there are few distinct values in the leading column of a composite index and many distinct values in the nonleading key of the index.

A reverse key index is a type of B-tree index that physically reverses the bytes of each index key while keeping the column order. For example, if the index key is 20, and if the two bytes stored for this key in hexadecimal are C1,15 in a standard B-tree index, then a reverse key index stores the bytes as 15,C1. Reversing the key solves the problem of contention for leaf blocks in the right side of a B-tree index.

In a bitmap index, the database stores a bitmap for each index key. In a conventional B-tree index, one index entry points to a single row. In a bitmap index, each index key stores pointers to multiple rows.

**Bitmap Join Index**

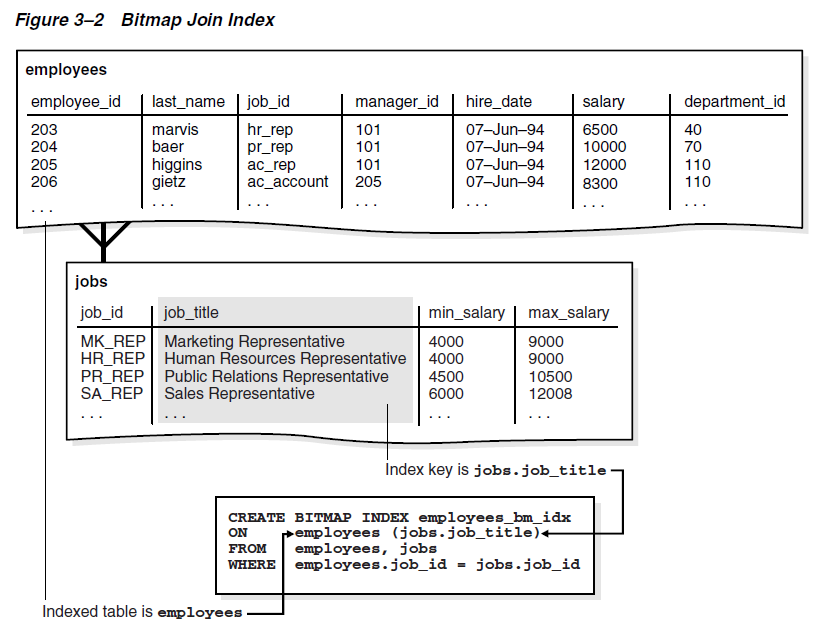
A bitmap join index is a bitmap index for the join of two or more tables. For each value in a table column, the index stores the rowid of the corresponding row in the indexed table.

An example when a bitmap join index would be useful:

SELECT COUNT(\*) FROM employees, jobs

WHERE employees.job\_id = jobs.job\_id

AND jobs.job\_title = 'Accountant';



jobs.job\_title employees.rowid

----------------------------------- ------------------

Accountant AAAQNKAAFAAAABSAAL

Accountant AAAQNKAAFAAAABSAAN

Accountant AAAQNKAAFAAAABSAAM

Accountant AAAQNKAAFAAAABSAAJ

Accountant AAAQNKAAFAAAABSAAK

Accounting Manager AAAQNKAAFAAAABTAAH

Administration Assistant AAAQNKAAFAAAABTAAC

Administration Vice President AAAQNKAAFAAAABSAAC

Administration Vice President AAAQNKAAFAAAABSAAB

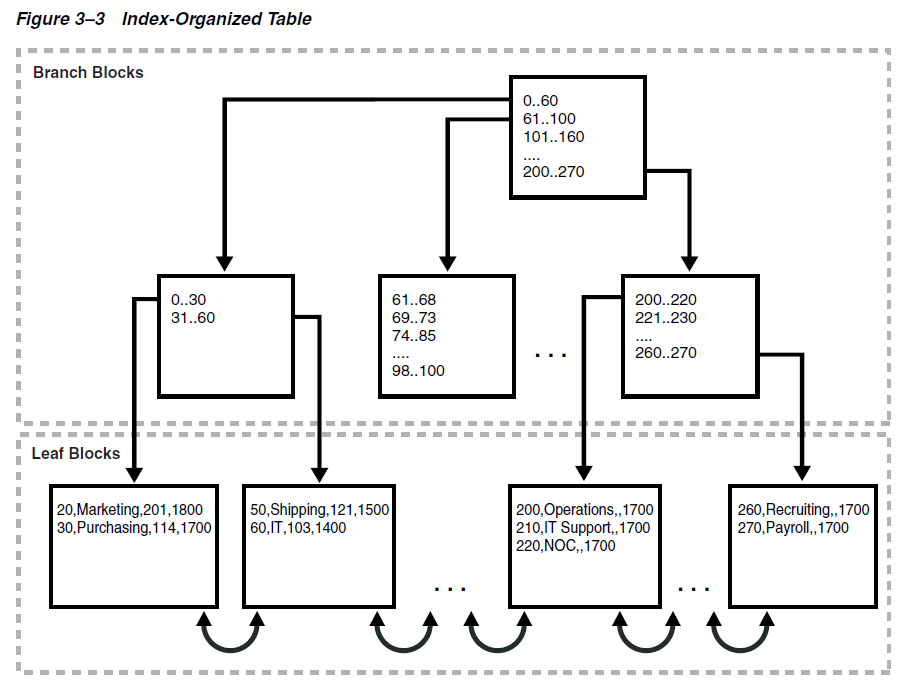
**Function-based Index**

You can create indexes on functions and expressions that involve one or more columns in the table being indexed. A function-based index computes the value of a function or expression involving one or more columns and stores it in the index. A function-based index can be either a B-tree or a bitmap index.

**Index-Organized Table**

An index-organized table is a table stored in a variation of a B-tree index structure. In a heap-organized table, rows are inserted where they fit. In an index-organized table, rows are stored in an index defined on the primary key for the table. Each index entry in the B-tree also stores the non-key column values.

An index-organized table stores all data in the same structure and does not need to store the rowid.



**View**

A view is a logical representation of one or more tables. In essence, a view is a stored query. A view derives its data from the tables on which it is based, called base tables. Base tables can be tables or other views. All operations performed on a view actually affect the base tables. You can use views in most places where tables are used.

Unlike a table, a view is not allocated storage space, nor does a view contain data. Rather, a view is defined by a query that extracts or derives data from the base tables referenced by the view. Because a view is based on other objects, it requires no storage other than storage for the query that defines the view in the data dictionary.

**Sequence**

A sequence is a schema object from which multiple users can generate unique integers. A sequence generator provides a highly scalable and well-performing method to generate surrogate keys for a number data type.

The sequence generator is useful in multiuser environments for generating unique numbers without the overhead of disk I/O or transaction locking. For example, two users simultaneously insert new rows into the orders table. By using a sequence to generate unique numbers for the order\_id column, neither user has to wait for the other to enter the next available order number. The sequence automatically generates the correct values for each user.

**Synonym**

A synonym is an alias for a schema object. For example, you can create a synonym for a table or view, sequence, PL/SQL program unit, user-defined object type, or another synonym. Because a synonym is simply an alias, it requires no storage other than its definition in the data dictionary.

Synonyms can simplify SQL statements for database users. Synonyms are also useful for hiding the identity and location of an underlying schema object. If the underlying object must be renamed or moved, then only the synonym must be redefined. Applications based on the synonym continue to work without modification.

**Data Dictionary**

The data dictionary base tables are the first objects created in any Oracle database. All data dictionary tables and views for a database are stored in the SYSTEM tablespace. Because the SYSTEM tablespace is always online when the database is open, the data dictionary is always available when the database is open.

The Oracle Database user SYS owns all base tables and user-accessible views of the data dictionary. Data in the base tables of the data dictionary is necessary for Oracle Database to function.

During database operation, Oracle Database reads the data dictionary to ascertain that schema objects exist and that users have proper access to them. Oracle Database also updates the data dictionary continuously to reflect changes in database structures, auditing, grants, and data.

For example, if user hr creates a table named interns, then new rows are added to the data dictionary that reflect the new table, columns, segment, extents, and the privileges that hr has on the table. This new information is visible the next time the dictionary views are queried.

Oracle Database creates public synonyms for many data dictionary views so users can access them conveniently

**Dynamic Performance Views**

Throughout its operation, Oracle Database maintains a set of virtual tables that record current database activity. These views are called dynamic performance views because they are continuously updated while a database is open and in use. The views, also sometimes called V$ views, contain information such as the following:

* System and session parameters
* Memory usage and allocation
* File states
* Progress of jobs and tasks
* SQL execution
* Statistics and metrics

Dynamic performance views are based on virtual tables built from database memory structures. Thus, they are not conventional tables stored in the database. Read consistency is not guaranteed for the views because the data is updated dynamically.

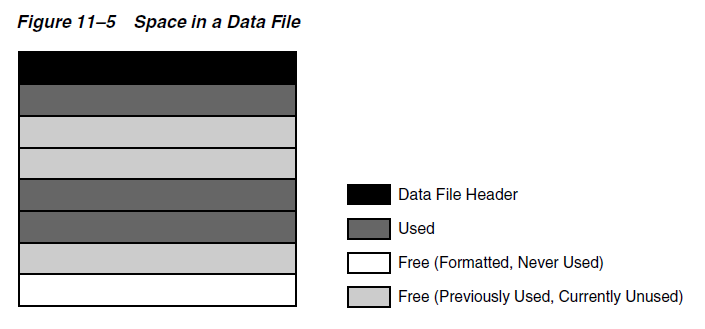
**Physical and Logical Storage**

An Oracle database is a set of files that store Oracle data in persistent disk storage.

A **data file** is a physical file on disk that was created by Oracle Database and contains data structures such as tables and indexes. A **temp file** is a data file that belongs to a temporary tablespace. The data is written to these files in an Oracle proprietary format that cannot be read by other programs.

Oracle Database creates a data file for a **tablespace** by allocating the specified amount of disk space plus the overhead for the data file header.

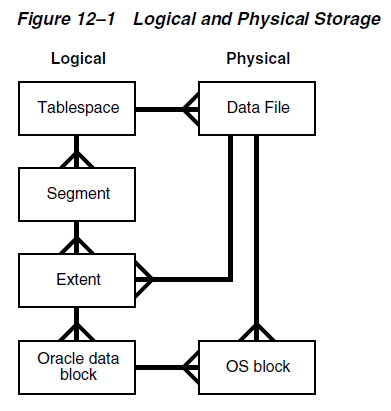
When Oracle Database first creates a data file, the allocated disk space is formatted but contains no user data. However, the database reserves the space to hold the data for future **segments** of the associated tablespace. As the data grows in a tablespace, Oracle Database uses the free space in the data files to allocate **extents** for the segment.



Oracle Database allocates logical space for all data in the database. The logical units of database space allocation are data blocks, extents, segments, and tablespaces. At a physical level, the data is stored in data files on disk. The data in the data files is stored in **operating system blocks**.

An operating system block is the minimum unit of data that the operating system can read or write. In contrast, an Oracle block is a logical storage structure whose size and structure are not known to the operating system.

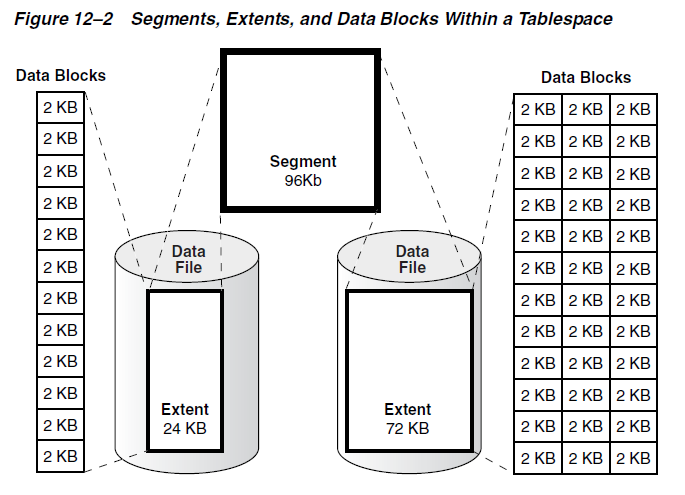
Figure 12–1 is an entity-relationship diagram for physical and logical storage. The crow's foot notation represents a one-to-many relationship.



At the finest level of granularity, Oracle Database stores data in **data blocks**. One logical data block corresponds to a specific number of bytes of physical disk space, for example, 2 KB. Data blocks are the smallest units of storage that Oracle Database can use or allocate.

An **extent** is a set of logically contiguous data blocks allocated for storing a specific type of information.

A **segment** is a set of extents allocated for a specific database object, such as a table. For example, the data for the employees table is stored in its own data segment, whereas each index for employees is stored in its own index segment. Every database object that consumes storage consists of a single segment.



**Logical Space Management**

Oracle Database must use logical space management to track and allocate the extents in a tablespace. When a database object requires an extent, the database must have a method of finding and providing it. Similarly, when an object no longer requires an extent, the database must have a method of making the free extent available.

Oracle Database manages space within a tablespace based on the type that you create. You can create either of the following types of tablespaces:

* Locally managed tablespaces (default)
* Dictionary-managed tablespaces

A **dictionary-managed tablespace** uses the data dictionary to manage its extents.

A **locally managed tablespace** maintains a bitmap in the data file header to track free and used space in the data file body. Each bit corresponds to a group of blocks. When space is allocated or freed, Oracle Database changes the bitmap values to reflect the new status of the blocks.

Within a locally managed tablespace, the database can manage segments automatically or manually.

**Automatic Segment Space Management** (ASSM) avoids the need to manually determine correct settings for many storage parameters. Only one crucial SQL parameter controls space allocation: **PCTFREE**. This parameter specifies the percentage of space to be reserved in a block for future updates.

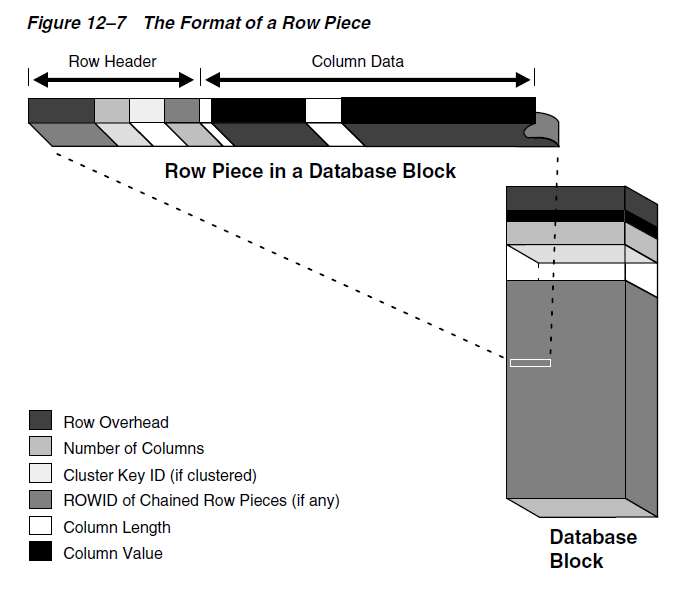
The legacy **Manual Segment Space Management** (MSSM) method uses a linked list called a free list to manage free space in the segment. In addition to PCTFREE, MSSM requires you to control space allocation with SQL parameters such as PCTUSED, FREELISTS, and FREELIST GROUPS.

**Block size**

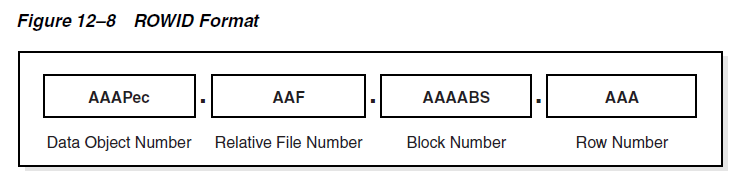
Every database has a database block size. The DB\_BLOCK\_SIZE initialization parameter sets the data block size for a database when it is created. The size is set for the SYSTEM and SYSAUX tablespaces and is the default for all other tablespaces. The database block size cannot be changed except by re-creating the database.

You can create individual tablespaces whose block size differs from the DB\_BLOCK\_SIZE setting.

Oracle Database stores rows as **variable-length records**. A row is contained in one or more row pieces. Each row piece has a row header and column data.

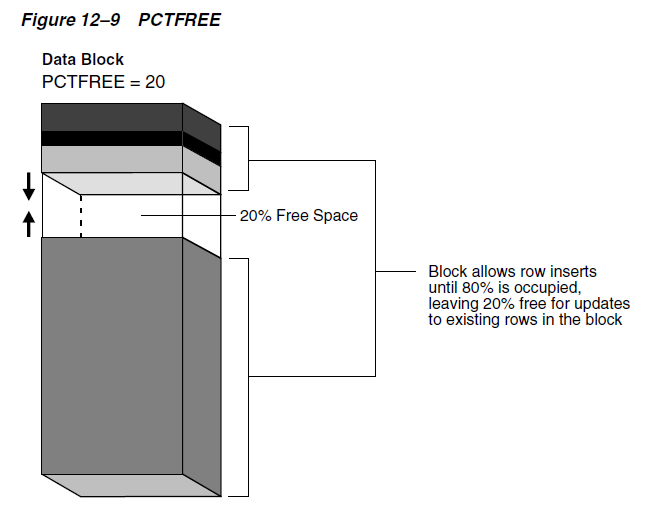


Oracle Database uses a **rowid** to uniquely identify a row. Internally, the rowid is a structure that holds information that the database needs to access a row. A rowid is not physically stored in the database, but is inferred from the file and block on which the data is stored.



As the database fills a data block from the bottom up, the amount of free space between the row data and the block header decreases. This free space can also shrink during updates, as when changing a trailing null to a nonnull value. The database manages free space in the data block to optimize performance and avoid wasted space.

The **PCTFREE** storage parameter is essential to how the database manages free space. This SQL parameter sets the minimum percentage of a data block reserved as free space for updates to existing rows.



A single data segment in a database stores the data for one user object. There are different types of segments. Examples of user segments include:

Table, table partition, table cluster, LOB or LOB partition, Index or index partition.

By default, the database uses **deferred segment creation** to update only database metadata when creating tables and indexes. When a user inserts the first row into a table or partition, the database creates segments for the table or partition, its LOB columns, and its indexes.