5.3 Table structures

Heap table

Row-oriented storage performs better than column-oriented storage for most transactional databases. Consequently, relational databases commonly use row-oriented storage. A ***table structure*** is a scheme for organizing rows in blocks on storage media.

Databases commonly support four alternative table structures:

- Heap table
- Sorted table
- Hash table
- Table cluster

Each table in a database can have a different structure. Databases assign a default structure to all tables. Database administrators can override the default structure to optimize performance for specific queries.

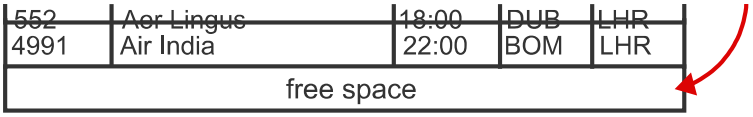
In a ***heap table***, no order is imposed on rows. The database maintains a list of blocks assigned to the table, along with the address of the first available space for inserts. If all blocks are full, the database allocates a new block and inserts rows in the new block.

When a row is deleted, the space occupied by the row is marked as free. Typically, free space is tracked as a linked list, as in the animation below. Inserts are stored in the first space in the list, and the head of the list is set to the next space.

Heap tables optimize insert, update, and delete operations. Heap tables are particularly fast for bulk load of many rows, since rows are stored in load order. Heap tables are not optimal for queries that read rows in a specific order, such as a range of primary key values, since rows are scattered randomly across storage media.

PARTICIPATION ACTIVITY 5.3.1: Heap table.

1 2 3 4 5 2x speed



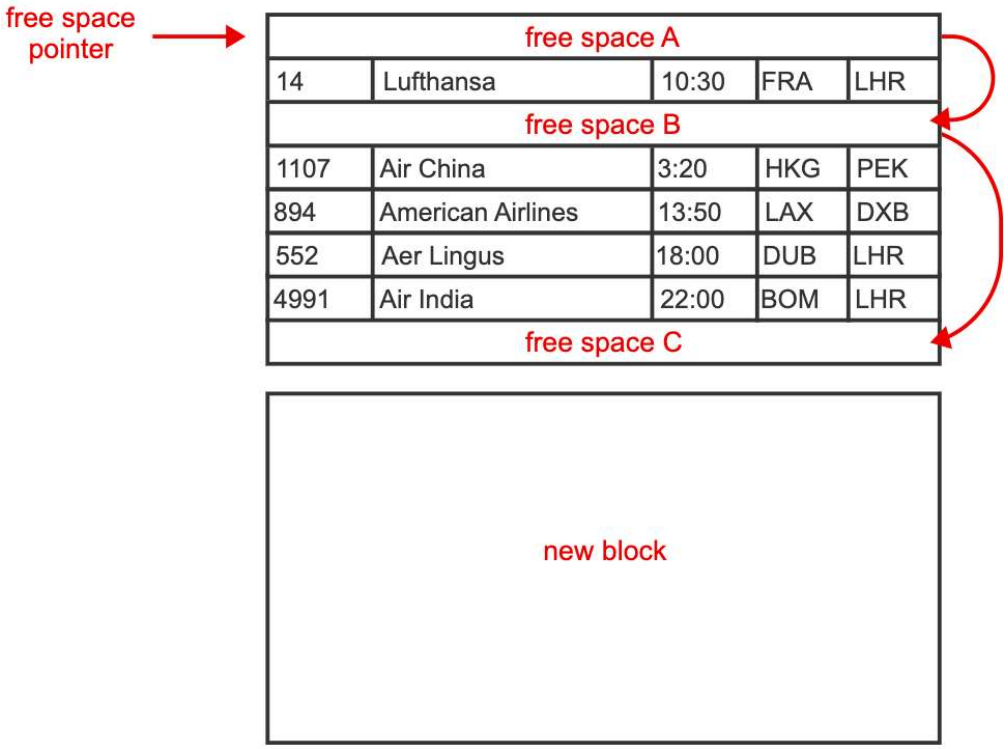
Inserts go to the first available space in the list.

Captions

1. In a heap table, the database maintains a pointer to free space, which indicates the location of the next insert.
2. When a row is inserted, the pointer moves to the next available space.
3. When a row is deleted, the pointer moves to the deleted space. The deleted space is linked to free space at the end of the table.
4. As more rows are deleted, free space is linked together in a list.
5. Inserts go to the first available space in the list.

Feedback?

PARTICIPATION ACTIVITY 5.3.2: Heap table.



Refer to the heap table above. Four new rows are inserted into the table. Match the free space to the insert.

Free space A	**First insert** The first inserted row is placed at the location pointed to by the free space pointer. The free space pointer is reset to B.	Correct
Free space B	**Second insert** After the first insert, the free space pointer points to B. Second insert is stored at free space B, and free space pointer is reset to space at the end of the block.	Correct
Free space C	**Third insert** After the second insert, the free space pointer points to C. Third insert is stored at C. Since the block is now full, a new block is allocated, and free space pointer points to the beginning of the new block.	Correct
New block	**Fourth insert** After the third insert, the block is full. A new block is allocated, and the fourth insert goes to the beginning of the new block.	Correct

Reset

Feedback?

Sorted table

In a ***sorted table***, the database designer identifies a ***sort column*** that determines physical row order. The sort column is usually the primary key but can be a non-key column or group of columns.

Rows are assigned to blocks according to the value of the sort column. Each block contains all rows with values in a given range. Within each block, rows are located in order of sort column values.

Sorted tables are optimal for queries that read data in order of the sort column, such as:

- JOIN on the sort column
- SELECT with range of sort column values in the WHERE clause

- SELECT with ORDER BY the sort column

Maintaining correct sort order of rows within each block can be slow. When a new row is inserted or when the sort column of an existing row is updated, free space may not be available in the correct location. To maintain the correct order efficiently, databases maintain pointers to the next row within each block, as in the animation below. With this technique, inserts and updates change two address values rather than move entire rows.

When an attempt is made to insert a row into a full block, the block splits in two. The database moves half the rows from the initial block to a new block, creating space for the insert.

In summary, sorted tables are optimized for read queries at the expense of insert and update operations. Since reads are more frequent than updates and inserts in many databases, sorted tables are often used, usually with the primary key as the sort column.

PARTICIPATION ACTIVITY 5.3.3: Sorted table.

1 2 3 4 2x speed

The sort order is maintained during an insert by changing two links rather than moving rows.

Captions

1. In a sorted table, rows are sorted on a sort column.
2. Inserting a new row requires moving all subsequent rows, which is inefficient.
3. To avoid inefficient inserts, the sort column order is maintained with links, producing a linked list.
4. The sort order is maintained during an insert by changing two links rather than moving rows.

Feedback?

PARTICIPATION ACTIVITY 5.3.4: Sorted table.

1) Assume 100 rows fit in each block of a sorted table. If a block is full and an insert causes the block to split, roughly what percentage of the new and old blocks is empty?

- ○ 33%
- ◉ 50%
- ○ 90%

Correct

When an insert causes a full block to split, roughly half the rows are reassigned to the new block. Both new and old blocks are roughly half full.

2) A database designer creates a new Employee table with primary key 'EmployeeNumber', loads 1 million rows into the table, then runs many queries that join other tables on the EmployeeNumber column. What is the best structure for Employee?

- ○ Sorted table
- ○ Heap table
- ◉ Heap table initially, then restructure to sorted table after bulk load is complete

Correct

Initially structuring the Employee table as a heap table optimizes for bulk load. Restructuring the table to a sorted table after loading all rows optimizes for join queries.

3) Sorted tables are always sorted on a single column.

- ○ True
- ◉ False

Correct

Sorted tables can be sorted on a group of columns. The first column in the group determines the physical order of rows. When the first column value is the same, then the second column determines physical order, and so on for additional columns.

Feedback?

Feedback?

Hash table

In a ***hash table***, rows are assigned to buckets. A ***bucket*** is a block or group of blocks containing rows. Initially, each bucket has one block. As a table grows, some buckets eventually fill up with rows, and the database allocates additional blocks. New blocks are linked to the initial block, and the bucket becomes a chain of linked blocks.

The bucket containing each row is determined by a hash function and a hash key. The ***hash key*** is a column or group of columns, usually the primary key. The ***hash function*** computes the bucket containing the row from the hash key.

Hash functions are designed to scramble row locations and evenly distribute rows across blocks. The ***modulo function*** is a simple hash function with four steps:

1. Convert the hash key by interpreting the key's bits as an integer value.
2. Divide the integer by the number of buckets.
3. Interpret the division remainder as the bucket number.
4. Convert the bucket number to the physical address of the block containing the row.

As tables grow, a fixed hash function allocates more rows to each bucket, creating deep buckets consisting of long chains of linked blocks. Deep buckets are inefficient since a query may read several blocks to access a single row. To avoid deep buckets, databases may use dynamic hash functions. A ***dynamic hash function*** automatically allocates more blocks to the table, creates additional buckets, and distributes rows across all buckets. With more buckets, fewer rows are assigned to each bucket and, on average, buckets contain fewer linked blocks.

Hash tables are optimal for inserts and deletes of individual rows, since row location is quickly determined from the hash key. For the same reason, hash tables are optimal for selecting a single row when the hash key value is specified in the WHERE clause. Hash tables are slow on queries that select many rows with a range of values, since rows are randomly distributed across many blocks.

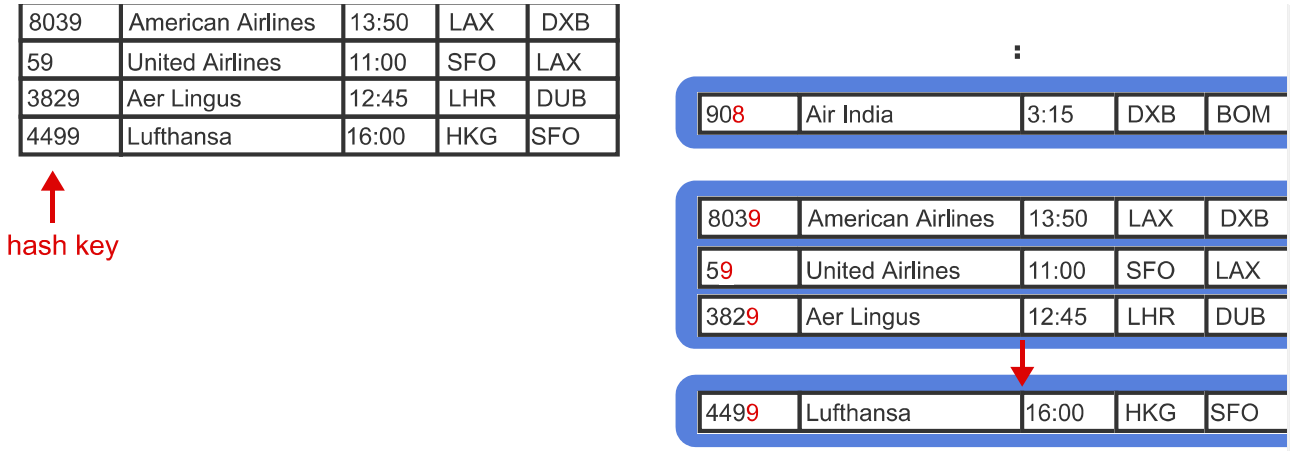
PARTICIPATION ACTIVITY 5.3.5: Assigning rows to buckets with a modulo function.

1 2 3 2x speed

Flight

Flight Number	AirlineName	Depart Time	Depart Airport	Arrive Airport
11	Lufthansa	10:30	FRA	LHR
552	Aer Lingus	18:00	DUB	LHR
908	Air India	3:15	DXB	BOM
940	American Airlines	5:05	ORD	SFO
1102	Air China	3:20	HKG	PEK

940	American Airlines	5:05	ORD	SFO
11	Lufthansa	10:30	FRA	LHR
552	Aer Lingus	18:00	DUB	LHR
1102	Air China	3:20	HKG	PEK



If a bucket fills with rows, additional blocks are allocated and linked to the first block.

Captions

1. FlightNumber is the hash key for the Flight table.
2. The hash function is modulo 10, resulting in 10 buckets. Rows are assigned to buckets based on the last digit of the hash key.
3. If a bucket fills with rows, additional blocks are allocated and linked to the first block.

Feedback?

PARTICIPATION ACTIVITY 5.3.6: Hash table.

Assume blocks are 8 kilobytes, rows are 200 bytes, and the hash function is modulo 13.

1) If each bucket initially has one block, how many rows fit into a single bucket?

- 13
- 40
- 520

Correct

Each bucket initially has one block, and each block is about 8,000 bytes. 8,000 bytes / 200 bytes/row = 40 rows per bucket.

2) In a best-case scenario, what is the maximum number of row inserts before some bucket overflows?

- 40

Correct

The hash function distributes rows to 13 buckets. Each bucket initially has one block, and each block can contain 8,000 bytes / 200 bytes/row = 40 rows. Therefore, up to 13 blocks × 40 rows/block = 520 rows can be loaded prior to overflow.

- 520
- Unlimited

3) 1,000,000 rows are inserted. In a worst-case scenario, what is the maximum number of blocks chained together in a single bucket?
   - 25,000
   - 100
   - 40

Correct

In a worst-case scenario, the value of the modulo 13 hash function is the same for all rows, so all rows are assigned to the same bucket. Since each block contains 40 rows, the bucket must contain 1,000,000 / 40 = 25,000 blocks.

4) What happens when a new row is inserted into a full bucket?
   - The full bucket splits in two, and half of the rows are moved to the new bucket.
   - A new block is linked to the initial block, and half the rows are moved from the initial block to the new block.
   - A new block is linked to the initial block, and the new row is stored in the new block.

Correct

When a row is inserted into a full bucket, a new block is allocated and linked to the initial block, creating space in the bucket for the new row.

Feedback?

CHALLENGE ACTIVITY 5.3.1: Table structures.

379958.2369558.qx3zqy7

Jump to level 1

1

2
3
4
5

Assume blocks are 10,800 bytes and rows are 90 bytes.

How many blocks would be required to fit 13,262 rows into a single bucket?

111

If only 1 block is used per bucket, how many rows in total can be added to 6 buckets?

720

1 2 3 4 **5**

Check Next **Done**. Click any level to practice more. Completion is preserved.

Expected: 111 blocks, 720 rows

(10,800 bytes/block) / (90 bytes/row) = 120 rows/block

So 13,262 rows would need:
13,262 rows / (120 rows/block) = 110 blocks with remainder 62, so 111 blocks.

Each bucket has 1 block, and 1 block has 120 rows. So 6 buckets can have up to 120 ×

Feedback?

Table clusters

Table clusters, also called ***multi-tables***, interleave rows of two or more tables in the same storage area. Table clusters have a ***cluster key***, a column that is available in all interleaved tables. The cluster key determines the order in which rows are interleaved. Rows with the same cluster key value are stored together. Usually the cluster key is the primary key of one table and the corresponding foreign key of another, as in the animation below.

Table clusters are optimal when joining interleaved tables on the cluster key, since physical row location is the same as output order. Table clusters perform poorly for many other queries:

- **Join on columns other than cluster key**. In a join on a column that is not the cluster key, physical row location is not the same as output order, so the join is slow.
- **Read multiple rows of a single table**. Table clusters spread each table across more blocks than other structures. Queries that read multiple rows may access more blocks.
- **Update clustering key**. Rows may move to different blocks when the clustering key changes.

Table clusters are not optimal for many queries and therefore are not commonly used.

PARTICIPATION ACTIVITY 5.3.7: Table cluster.

1 2 3 2x speed

Airline

AirlineName	Code	Country
Lufthansa	LH	Germany
Aer Lingus	EI	Ireland
Air India	AI	India
American Airlines	AA	United States
Air China	CA	China

Flight

Flight Number	AirlineName	Depart Time	Depart Airport	Arrive Airport
11	Lufthansa	10:30	FRA	LHR
552	Aer Lingus	18:00	DUB	LHR
908	Lufthansa	3:15	DXB	BOM
940	American Airlines	5:05	ORD	SFO
1102	Air China	3:20	HKG	PEK
8039	American Airlines	13:50	LAX	DXB

Lufthansa	LH	Germany		
11	Lufthansa	10:30	FRA	LHR
908	Lufthansa	3:15	DXB	BOM
Aer Lingus	EI	Ireland		
552	Aer Lingus	18:00	DUB	LHR
Air India	AI	India		
American Airlines	AA	United States		
940	American Airlines	5:05	ORD	SFO
8039	American Airlines	13:50	LAX	DXB
Air China	CA	China		
1102	Air China	3:20	HKG	PEK

Rows of Airline and Flight are interleaved on storage media.

Captions

1. Airline and Flight tables both contain an AirlineName column.
2. AirlineName is the cluster key for the table cluster.
3. Rows of Airline and Flight are interleaved on storage media.

Feedback?

PARTICIPATION ACTIVITY 5.3.8: Table clusters.

1) The cluster key is normally the primary key of both tables in a cluster.
   - True
   - False (selected)

Correct

The cluster key is often the primary key of one table and the corresponding foreign key of another. This optimizes primary key - foreign key join queries.

2) Table clusters are the most commonly used physical structure for tables.
   - True
   - False (selected)

Correct

Table clusters are optimal for joins on the clustering key, but not for most other queries. Since table clusters perform poorly for many types of queries, table clusters are not commonly used.

3) Cluster keys in a table cluster are similar to sort columns in a sorted table.
   - True (selected)
   - False

Correct

Cluster keys are similar to sort columns, in that both determine the physical order of rows on storage media.

4) Table clusters always contain exactly two tables.
   - True
   - False (selected)

Correct

Table clusters can contain three or more tables, as long as all tables share a common cluster key column. However, clusters of three or more tables are uncommon.

Feedback?

How

was this section? Provide feedback