16.1 Other indexes

Hash indexes

The multi-level index is the most commonly used index type. Several additional index types are used less often but supported by many databases:

- Hash index
- Bitmap index
- Logical index
- Function index

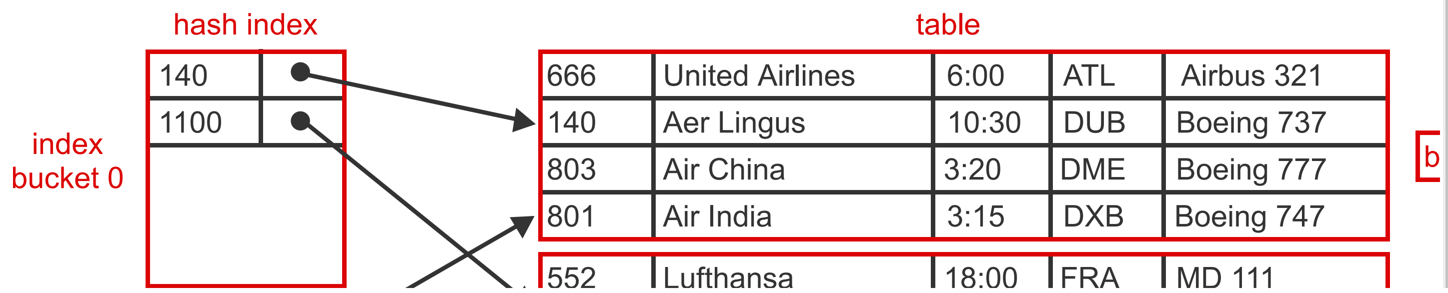
In a ***hash index***, index entries are assigned to buckets. A ***bucket*** is a block or group of blocks containing index entries. Initially, each bucket has one block. As an index grows, some buckets eventually fill up, and additional blocks are allocated and linked to the initial block.

The bucket containing each index entry is determined by a ***hash function***, which computes a bucket number from the value of the indexed column. To locate a row containing a column value, the database:

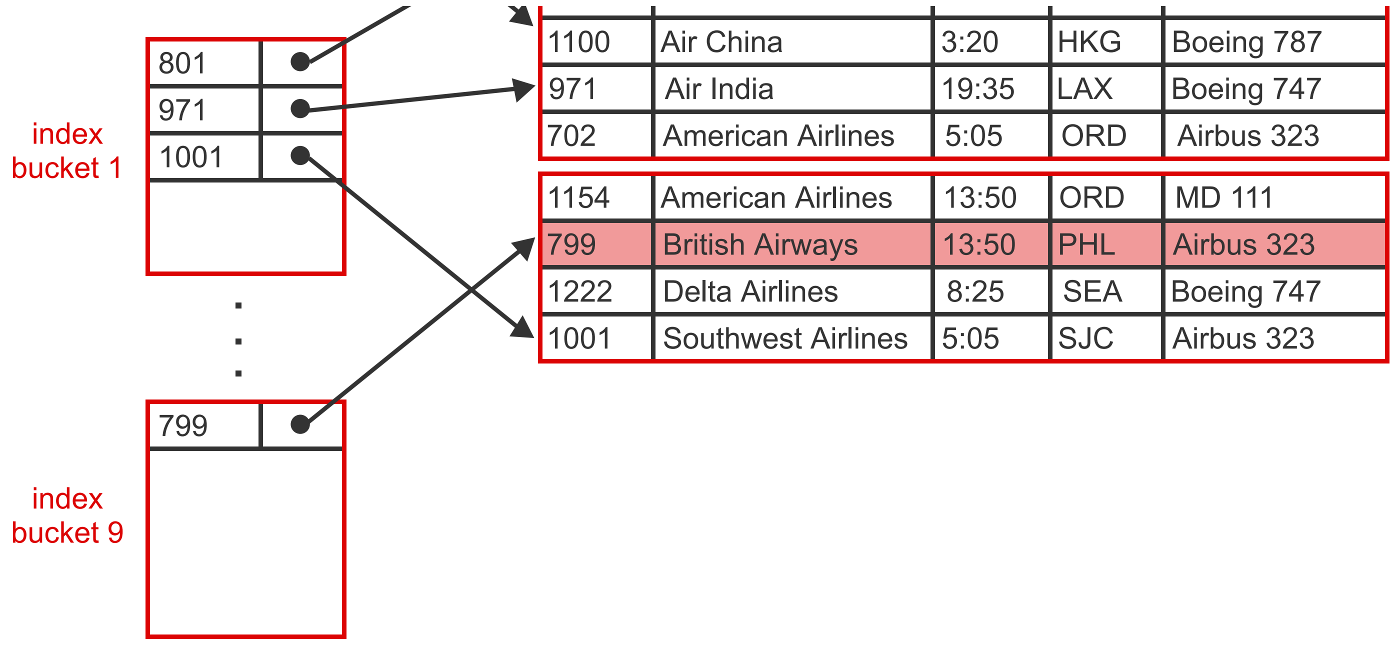
1. Applies the hash function to the column value to compute a bucket number.
2. Reads the index blocks for the bucket number.
3. Finds the index entry for the column value and reads the table block pointer.
4. Reads the table block containing the row.

A hash index is similar to a hash table, described in another section. However, a hash index stores *index entries* in each bucket, while a hash table stores *table rows* in each bucket.

PARTICIPATION ACTIVITY 16.1.1: Hash index.



666	United Airlines	6:00	ATL	Airbus 321
140	Aer Lingus	10:30	DUB	Boeing 737
803	Air China	3:20	DME	Boeing 777
801	Air India	3:15	DXB	Boeing 747
552	Lufthansa	18:00	FRA	MD 111



index bucket 1
801
971
1001
799
index bucket 9
1100 Air China 3:20 HKG Boeing 787
971 Air India 19:35 LAX Boeing 747
702 American Airlines 5:05 ORD Airbus 323
1154 American Airlines 13:50 ORD MD 111
799 British Airways 13:50 PHL Airbus 323
1222 Delta Airlines 8:25 SEA Boeing 747
1001 Southwest Airlines 5:05 SJC Airbus 323

Animation content:

Static figure:
A sorted table has three blocks. Each block contains four rows. Each row contains flight number, airline name, departure time, airport code, and aircraft type. The rows are sorted on airport code.

A hash index appears with buckets 0, 1, and 9. Buckets 2 through 8 are not shown. Each bucket has entries with a flight number and pointer to the table block containing the corresponding row. The last digits of all flight numbers in each hash bucket are the same as the bucket number.

Step 1: The table is clustered on DepartureAirportCode. The table appears. Airport codes are highlighted.

Step 2: The hash index is on FlightNumber. The hash function is modulo 10. Flight numbers are highlighted.

Step 3: Index entries for flight numbers ending in 0 go in bucket 0. Bucket 0 appears. All flight numbers in the index entries end in 0. Index entries point to table rows containing the corresponding flight number.

Step 4: Index entries for flight numbers ending in 1 go in bucket 1. Bucket 1 appears. All flight numbers in the index entries end in 1. Index entries point to table rows containing the corresponding flight number.

Step 5: Since the hash function is modulo 10, the hash index has 10 buckets. Bucket 9 appears. All flight numbers in the index entries end in 9. Index entries point to table rows containing the corresponding flight number.

Animation captions:

1. The table is clustered on DepartureAirportCode.
2. The hash index is on FlightNumber. The hash function is modulo 10.
3. Index entries for flight numbers ending in 0 go in bucket 0.
4. Index entries for flight numbers ending in 1 go in bucket 1.
5. Since the hash function is modulo 10, the hash index has 10 buckets.

Terminology

*Sometimes the term hash index is used to mean a hash key, but the two terms are different. A **hash index** is an index that is structured using a hash function. A **hash key** is a column that determines the physical location of rows in a hash table.*

PARTICIPATION ACTIVITY | 16.1.2: Hash indexes.

1) A hash table can have a hash index.

- True
- False

2) A primary index can be structured as a hash index.

- True
- False

3) A hash key can be structured as a multi-level index.

- True
- False

4) A hash index can be sparse.

- True
- False

Bitmap indexes

A ***bitmap index*** is a grid of bits:

- Each index row corresponds to a unique table row. If the table's primary key is an integer, the index row number might be the primary key value. Alternatively, the index row number might be an internal table row number, maintained by the database.
- Each index column corresponds to a distinct table value. Ex: If the index is on AirportCode, each index column corresponds to a different three-letter airport code. The mapping of index column numbers to table values is computed with a function or stored in an internal 'lookup' table.

Bitmap indexes contain ones and zeros. 'One' indicates that the table row corresponding to the index row number contains the table value corresponding to the index column number. 'Zero' indicates the row does not contain the value.

To locate rows containing a table value, the database:

1. Determines the index column corresponding to the table value.
2. Reads the index column and finds index rows that are set to 'one'.
3. Determines table rows corresponding to the index rows.
4. Determines pointers to blocks containing the table rows.
5. Reads the blocks containing the table rows.

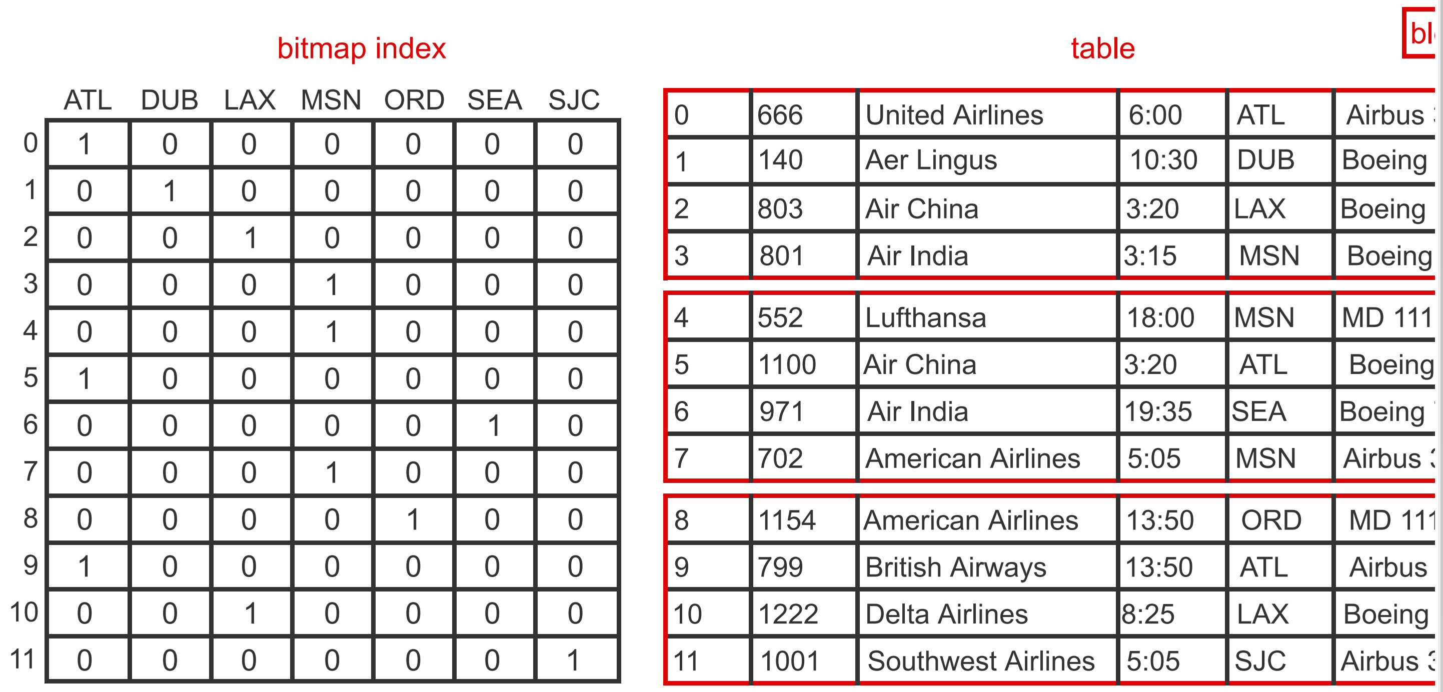
An efficient bitmap index has two characteristics:

- The database can quickly determine the block containing a table row from the index row number (steps 3 and 4). Ex: The index row number is the hash key for a hash table. The block is determined by applying the hash function to the row number. Ex: The index row number is the table primary key. The block is determined with a primary index.
- The indexed column contains relatively few distinct values, typically tens or hundreds. If the indexed column contains thousands of distinct values, the bitmap index is large and inefficient.

Bitmap indexes with the above characteristics enable fast reads. Ex: A table with 10 million rows has a bitmap index on a column with 100 distinct values. The index contains 125 million bytes (10 million rows × 100 column values × 1 bit per index entry / 8 bits per byte) and can easily be retained in memory.

PARTICIPATION ACTIVITY

16.1.3: Bitmap index.



	ATL	DUB	LAX	MSN	ORD	SEA	SJC
0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0
2	0	0	1	0	0	0	0
3	0	0	0	1	0	0	0
4	0	0	0	1	0	0	0
5	1	0	0	0	0	0	0
6	0	0	0	0	0	1	0
7	0	0	0	1	0	0	0
8	0	0	0	0	1	0	0
9	1	0	0	0	0	0	0
10	0	0	1	0	0	0	0
11	0	0	0	0	0	0	1

0	666	United Airlines	6:00	ATL	Airbus
1	140	Aer Lingus	10:30	DUB	Boeing
2	803	Air China	3:20	LAX	Boeing
3	801	Air India	3:15	MSN	Boeing
4	552	Lufthansa	18:00	MSN	MD 111
5	1100	Air China	3:20	ATL	Boeing
6	971	Air India	19:35	SEA	Boeing
7	702	American Airlines	5:05	MSN	Airbus
8	1154	American Airlines	13:50	ORD	MD 11
9	799	British Airways	13:50	ATL	Airbus
10	1222	Delta Airlines	8:25	LAX	Boeing
11	1001	Southwest Airlines	5:05	SJC	Airbus

Animation content:

Static figure:
A sorted table has three blocks. Each block contains four rows. Each row contains row number, flight number, airline name, departure time, airport code, and aircraft type. Rows are numbered 0 through 11 and are sorted on row number.

A bitmap index appears. The index is a grid. Each grid cell contains 0 or 1. Grid rows are numbered 0 through 11. Grid columns are labeled with airport codes that appear in the table: ATL, DUB, LAS, MSN, ORD, SEA, and SJC.

Step 1: The database maintains row numbers internally. The table appears. Row numbers are highlighted.

Step 2: A bitmap index is a grid of bits. Bitmap index rows correspond to internal table row numbers. The bitmap index appears with no values in the grid. Row numbers appear.

Step 3: Bitmap index columns correspond to table values. Grid column labels appear.

Step 4: Table row 0 contains ATL, so the corresponding index bit is 1. 1 appears in grid cell (0, ATL). Table row 0 contains airport code ATL and is highlighted.

Step 5: Table row 1 contains DUB, so the corresponding index bit is 1. 1 appears in grid cell (1, DUB). Table row 1 contains airport code DUB and is highlighted.

Step 6: Each 1 in the bitmap index indicates the corresponding value appears in the table row. For each row in the table, a 1 appears in the grid cell for the row's number and airport code.

Step 7: Each 0 in the bitmap index indicates the corresponding value does not appear in the table row. 0 appears in the remaining grid cells.

Animation captions:

1. The database maintains row numbers internally.
2. A bitmap index is a grid of bits. Bitmap index rows correspond to internal table row numbers.
3. Bitmap index columns correspond to table values.
4. Table row 0 contains ATL, so the corresponding index bit is 1.
5. Table row 1 contains DUB, so the corresponding index bit is 1.
6. Each 1 in the bitmap index indicates the corresponding value appears in the table row.
7. Each 0 in the bitmap index indicates the corresponding value does not appear in the table row.

PARTICIPATION ACTIVITY 16.1.4: Bitmap indexes.

1) Refer to the bitmap index in the above animation. The three 1's in the MSN column means:

- ○ The value MSN appears in three table blocks.
- ○ The value MSN appears in three table rows.
- ○ The value MSN appears in three buckets.

2) How many 1's can appear in a single row of a bitmap index?

- ○ Exactly one

- ○ Zero or one
- ○ One or many

3) California has 26 million licensed drivers and 2,597 ZIP codes. The California Department of Motor Vehicles tracks one address for every licensed driver in a table with a ZipCode column. How many bytes are in a bitmap index on ZipCode?

- ○ Approximately 26 million
- ○ Exactly 2,597 / 8
- ○ Approximately 8.4 billion

Logical indexes

A single- or multi-level index normally contains pointers to table blocks and is called a ***physical index***.

A ***logical index*** is a single- or multi-level index in which pointers to table blocks are replaced with primary key values. Logical indexes are always secondary indexes and require a separate primary index on the same table. To locate a row containing a column value, the database:

1. Looks up the column value in the logical index to find the primary key value.
2. Looks up the primary key value in the primary index to find the table block pointer.
3. Reads the table block containing the row.

Logical indexes change only when primary key values are updated, which occurs infrequently. Physical indexes change whenever a row moves to a new block, which occurs in several ways:

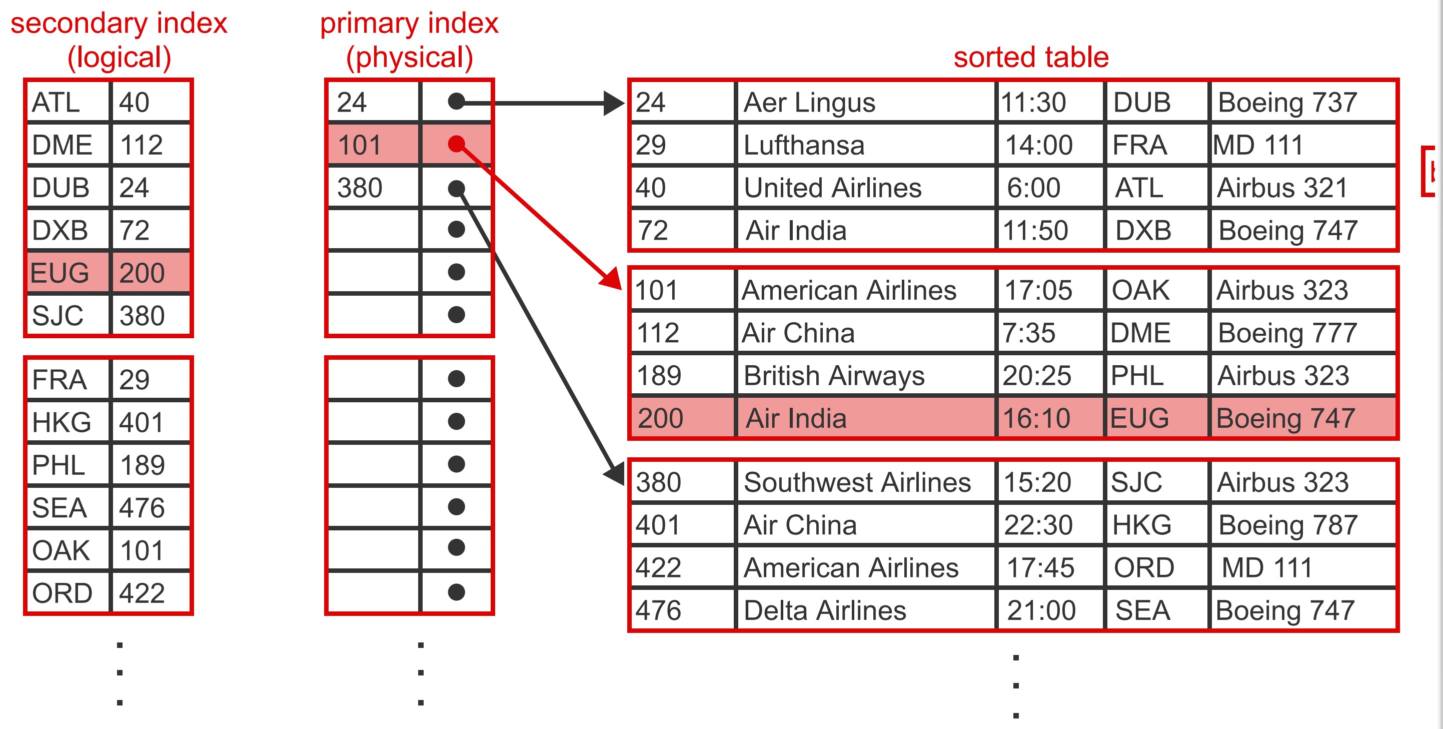
- *A row is inserted into a full block*. To create space for the new row, the block splits and some rows move to a new block.
- *The sort column is updated*. When the sort column is updated, the row may move to a new block to maintain sort order.
- *The table is reorganized*. Occasionally, a database administrator may physically reorganize a table to recover deleted space or order blocks contiguously on magnetic disk.

If a table has several indexes, the time required to update physical indexes is significant, and logical indexes are more efficient.

On read queries, a logical index requires an additional read of the primary index and is slower than a

On read queries, a logical index requires an additional read of the primary index and is slower than a physical index. However, the primary index is often retained in memory, mitigating the cost of the additional read.

PARTICIPATION ACTIVITY 16.1.5: Logical index.



secondary index (logical)	
ATL	40
DME	112
DUB	24
DXB	72
EUG	200
SJC	380
FRA	29
HKG	401
PHL	189
SEA	476
OAK	101
ORD	422

primary index (physical)	
24	●
101	●
380	●

sorted table				
24	Aer Lingus	11:30	DUB	Boeing 737
29	Lufthansa	14:00	FRA	MD 111
40	United Airlines	6:00	ATL	Airbus 321
72	Air India	11:50	DXB	Boeing 747
101	American Airlines	17:05	OAK	Airbus 323
112	Air China	7:35	DME	Boeing 777
189	British Airways	20:25	PHL	Airbus 323
200	Air India	16:10	EUG	Boeing 747
380	Southwest Airlines	15:20	SJC	Airbus 323
401	Air China	22:30	HKG	Boeing 787
422	American Airlines	17:45	ORD	MD 111
476	Delta Airlines	21:00	SEA	Boeing 747

Animation content:

Static figure:
A sorted table has three blocks. Each block contains four rows. Each row contains flight number, airline name, departure time, airport code, and aircraft type. The table is sorted on fight number.

A sparse index appears, with caption primary index (physical). The index contains one level. Each entry contains a flight number and pointer to the table block that begins with the corresponding row.

A dense index appears, with caption secondary index (logical). The index contains one level. Each entry contains an airport code and a flight number. Every airport code in the table appears in this index, along with the flight number for the corresponding row.

Step 1: The table is sorted on FlightNumber, the primary key. This example assumes FlightNumber is unique across all airlines. The sorted table appears. Flight numbers are

FlightNumber is unique across all airlines. The sorted table appears. Flight numbers are highlighted.

Step 2: The table has primary index on FlightNumber. The primary index is physical and sparse. The primary index appears.

Step 3: The table has secondary index on DepartureAirportCode. The secondary index is logical and dense. The secondary index appears.

Step 4: To find the row containing EUG, the database finds the primary key 200 in the secondary index, then finds the block pointer in the primary index leading to the block containing 200. In the secondary index, entry (EUG, 200) is highlighted. Next to the primary index, 200 appears between entries for 101 and 380. Primary index entry 101 is highlighted and points to the table block beginning with flight number 101. The last row of this table block contains flight number 200 and is highlighted.

Animation captions:

1. The table is sorted on FlightNumber, the primary key. This example assumes FlightNumber is unique across all airlines.
2. The table has primary index on FlightNumber. The primary index is physical and sparse.
3. The table has secondary index on DepartureAirportCode. The secondary index is logical and dense.
4. To find the row containing EUG, the database finds the primary key 200 in the secondary index, then finds the block pointer in the primary index leading to the block containing 200.

PARTICIPATION ACTIVITY | 16.1.6: Logical indexes.

If unable to drag and drop, refresh the page.

Secondary index **Primary index** **Physical index** **Logical index**

	An index on a unique sort column.
	An index on a non-sort column.
	An index with primary key values rather than block pointers.

An index with table block pointers.

Reset

Function indexes

In some cases, values specified in a WHERE clause may be in a different format or units than values stored in the column. Ex:

- The WHERE clause specifies values in upper case, but the column contains mixed upper and lower case characters.
- The WHERE clause specifies values as percentages, from 0 to 100, but the column contains values from 0 to 1.

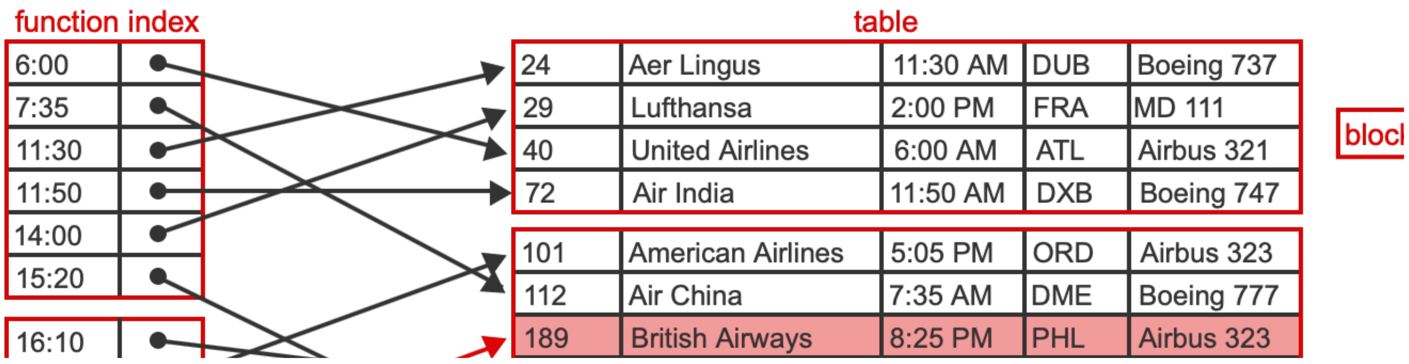
In the above examples, index entries do not match values in the WHERE clause, so the database cannot use the index to execute the query.

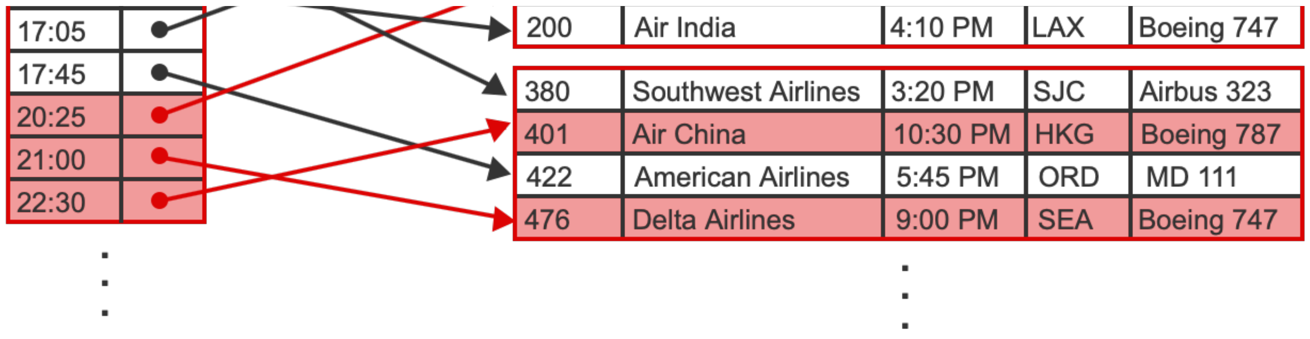
To address this problem, some databases support function indexes. In a ***function index***, the database designer specifies a function on the column value. Index entries contain the result of the function applied to column values, rather than the column values.

Ex: Column values are stored as decimal numbers between 0 and 1, but users specify percentages as integers between 0 and 100 in queries. The database designer specifies a function index that multiplies column values by 100 and converts the result to an integer. The index contains integers between 0 and 100, so the database can use the function index to process queries.

In principle, functions can be used with any index type, including single-level, multi-level, hash, bitmap, and logical indexes. In practice, support varies by database.

PARTICIPATION ACTIVITY 16.1.7: Function index.


function index
6:00
7:35
11:30
11:50
14:00
15:20
16:10
table
24 | Aer Lingus | 11:30 AM | DUB | Boeing 737
29 | Lufthansa | 2:00 PM | FRA | MD 111
40 | United Airlines | 6:00 AM | ATL | Airbus 321
72 | Air India | 11:50 AM | DXB | Boeing 747
101 | American Airlines | 5:05 PM | ORD | Airbus 323
112 | Air China | 7:35 AM | DME | Boeing 777
189 | British Airways | 8:25 PM | PHL | Airbus 323
block


17:05
17:45
20:25
21:00
22:30
200 | Air India | 4:10 PM | LAX | Boeing 747
380 | Southwest Airlines | 3:20 PM | SJC | Airbus 323
401 | Air China | 10:30 PM | HKG | Boeing 787
422 | American Airlines | 5:45 PM | ORD | MD 111
476 | Delta Airlines | 9:00 PM | SEA | Boeing 747

```sql
SELECT FlightNumber, AirlineName
FROM Flight
WHERE DepartureTime > "20:00"
```

Animation content:

Static figure:
A sorted table has three blocks. Each block contains four rows. Each row contains flight number, airline name, departure time, airport code, and aircraft type. The table is sorted on fight number.

A dense index appears with caption function index. Index entries contain a time and pointer to the corresponding table row. Time in the index is formatted on a 24-hour clock. Ex: 16:10. Time in the table is formatted AM/PM. Ex: 4:10 PM.

An SQL statement appears.
Begin SQL code:
SELECT FlightNumber, AirlineName
FROM Flight
WHERE DepartureTime > '20:00';
End SQL code.

Step 1: The table contains times in AM/PM format. The table appears. Time values are highlighted.

Step 2: Queries specify time in 24-hour format. The SQL statement appears. The WHERE clause is highlighted.

Step 3: The function index converts AM/PM format to 24-hour format. Each index entry, along with the corresponding table row, is highlighted.

Step 4: The function index format is consistent with the WHERE clause, so the database can use the index.

Animation captions:

1. The table contains times in AM/PM format.
2. Queries specify time in 24-hour format.
3. The function index converts AM/PM format to 24-hour format.
4. The function index format is consistent with the WHERE clause, so the database can use the index.

PARTICIPATION ACTIVITY 16.1.8: Function indexes.

1) Refer to the animation above. Which table blocks does the following query read?

```
SELECT FlightNumber,
AirlineName
FROM Flight
WHERE DepartureTime > "12:00"
AND DepartureTime < "16:00";
```

- ○ Block 0
- ○ Block 1
- ○ Blocks 0 and 2

2) Which index types can also be function indexes?

- ○ Only function index types
- ○ All index types except bitmap
- ○ Hash, bitmap, single-level, multi-level, and logical index types

3) When should a database designer consider a function index?

- ○ The display format is different than storage format
- ○ The WHERE clause format is different than storage format

○ The input format is different than storage format

CHALLENGE ACTIVITY | 16.1.1: Other indexes.

544874.3500394.qx3zqy7

Start

Given that the hash index is on FlightNumber, and the hash function is modulo 10, select t
row(s) with a hash index in bucket 1.

table

☐	939	United Airlines	6:00	ATL	Airbus 321
☐	257	Aer Lingus	10:30	DUB	Boeing 737
☐	721	Air China	3:20	DME	Boeing 777
☐	318	Air India	3:15	DXB	Boeing 747

☐	911	Lufthansa	18:00	FRA	MD 111
☐	533	Air China	3:20	HKG	Boeing 787
☐	601	Air India	19:35	LAX	Boeing 747
☐	892	American Airlines	5:05	ORD	Airbus 323

☐	764	American Airlines	13:50	ORD	MD 111
☐	909	British Airways	13:50	PHL	Airbus 323
☐	877	Delta Airlines	8:25	SEA	Boeing 747
☐	113	Southwest Airlines	5:05	SJC	Airbus 323

1 | 2 | 3 | 4

Check Next