Index for Employee

Part of EMPLOYEE table:

	employee_id	first_name	last_name	job_id	salary	manager_id	department_id
1	100	Steven	King	AD_PRES	24000.00	NULL	90
2	101	Neena	Kochhar	AD_VP	17000.00	100	90
3	102	Lex	De Haan	AD_VP	17000.00	100	90
4	103	Alexander	Hunold	IT_PROG	9000.00	102	60
5	104	Bruce	Ernst	IT_PROG	6000.00	103	60
6	107	Diana	Lorentz	IT_PROG	4200.00	103	60
7	124	Kevin	Mourgos	ST_MAN	5800.00	100	50
8	141	Trenna	Rajs	ST_CLERK	3500.00	124	50
9	142	Curtis	Davies	ST_CLERK	3100.00	124	50
10	143	Randall	Matos	ST_CLERK	2600.00	124	50
11	144	Peter	Vargas	ST_CLERK	2500.00	124	50
12	149	Eleni	Zlotkey	SA_MAN	10500.00	100	80
13	174	Ellen	Abel	SA_REP	11000.00	149	80
14	176	Jonathon	Taylor	SA_REP	8600.00	149	80
15	178	Kimberely	Grant	SA_REP	7000.00	149	NULL
16	200	Jennifer	Whalen	AD_ASST	4400.00	101	10
17	201	Michael	Hartstein	MK_MAN	13000.00	100	20
18	202	Pat	Fay	MK_REP	6000.00	201	20
19	205	Shelley	Higgins	AC_MGR	12000.00	101	110
20	206	William	Gietz	AC_ACCOUNT	8300.00	205	110

Query:

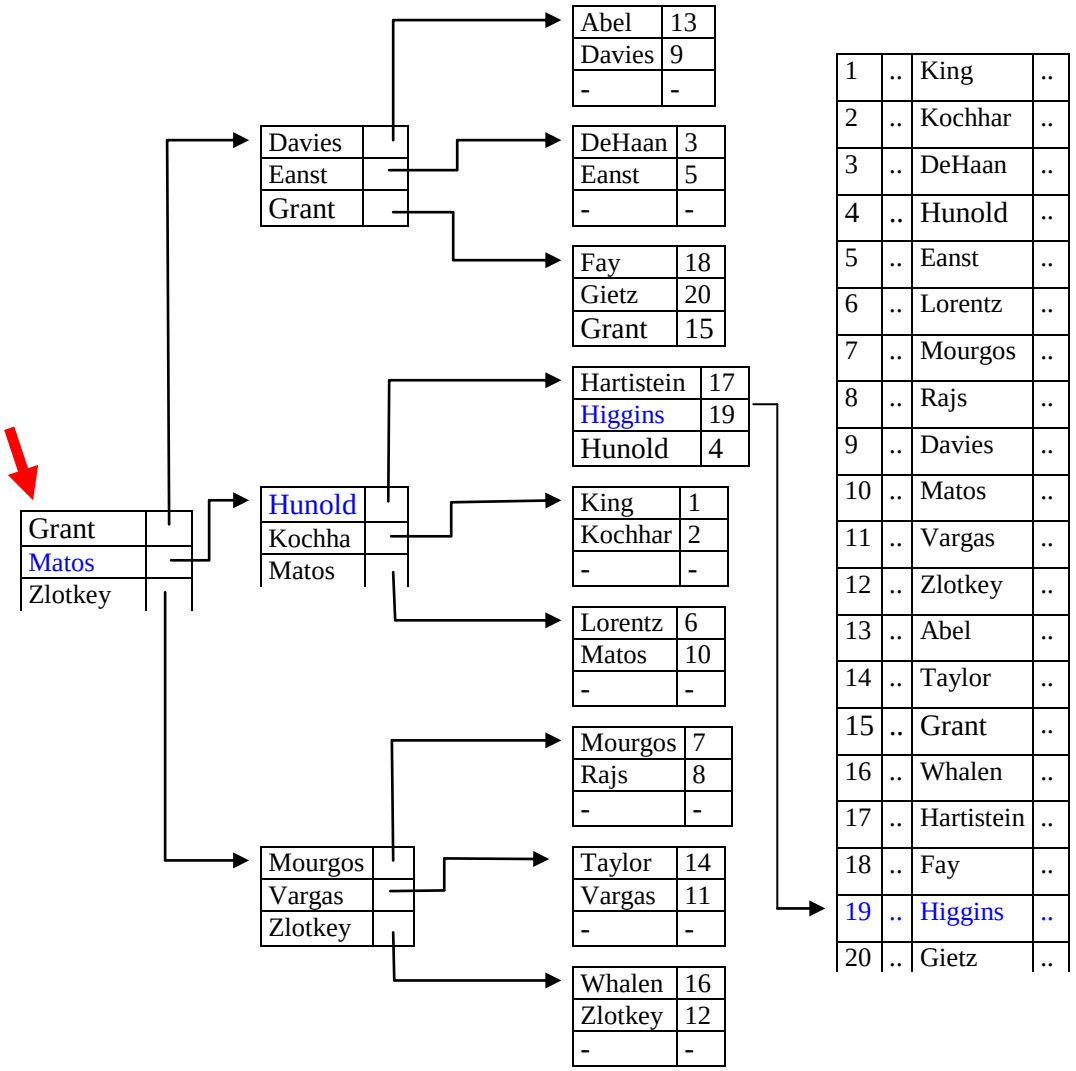
```
SELECT employee_id, salary
FROM  EMPLOYEES
WHERE last_name ='Higgins';
```

SQL Server scans full table even if the row was found in the middle of the table (as DBMS doesn't know if another row exists).

After creating an INDEX on the column, say

```
CREATE INDEX emp_last_name_index
ON EMPLOYEE (last_name);
```

DBMS created a structure as below, and following query based on last_name will be much quicker as DBMS will check the index rather than scan the whole table.



1. To search for ‘Higgins’, Searching starts from the root of the B-tree, A search proceeds for a value greater than or equal to the value to be retrieved (i.e., ‘Higgins’ in this case)

(1) From the root, the value ‘Matos’ is retrieved (as ‘Higgins’> ‘Grant’; ‘Higgins’<= ‘Matos’), which leads to the node in middle of the 2nd level;
(2) From the middle node of the second level of the B-tree, ‘Hunold’ is retrieved (as ‘Higgins’<= ‘Hunold’). Go to third level; Go to its first node in the third level;
(3) ‘Higgins’ is found in the node, and from the Row ID (i.e., 19 as simplified), SQL server can locate the row from the data table.

2. All value can be found from the B-tree index, or reported ‘miss’ (if it is not in) in no more than three sets of comparison. For example, search for ‘Zugarbi’ will lead to a ‘miss’ after one comparison.

3. Real row ID is much complicated, a value combined the, filegroupID, database ID, data block ID and the Row Id within the data block.

	last_name	Row Pointer
1	Abel	710D366B-A709-4D34-A07D-E919DBCEA6E2
2	Davies	3854DB14-1A0B-451F-900A-7CFC80E43C34
3	De Haan	68FE87C3-23EC-4FCB-BF50-78C410AB4F76
4	Emst	81A957D5-1011-45F6-9847-ED82285B6613
5	Fay	721C9C4C-FE50-4B35-9523-988086162EF0
6	Gietz	DFE2618E-A508-415E-88C6-CA0A9AA3A479
7	Grant	09833612-0A16-4FB0-95CD-83100B69580E
8	Hartstein	91F6AF31-3D0E-44E8-827D-B87F553A1619
9	Higgins	3D06EEDE-7591-4916-B7B9-6EF96E0FE88D
10	Hunold	6AE12988-B4C5-4555-8600-E03584F9026B
11	King	A2CDDA9A-ED42-46E9-B948-A25D61E771C6
12	Kochhar	FE8886E9-7B7A-4F42-A681-442D64EACFA8
13	Lorentz	3417BF3D-B865-42A7-9298-2735581350E9
14	Matos	B915DAB8-8831-453C-8277-A1A9A09E32C5
15	Mourgos	0E3EAE96-F823-49AC-9C22-6FE4884A3370
16	Rajs	E24E2B7B-0AA1-4E4D-8042-B060532A7E1B
17	Taylor	68BDCB5D-7214-40DC-9C3B-026D6B67394B
18	Vargas	D9927686-745B-4E98-9A1B-1A1901728F76
19	Whalen	729FA093-BAD6-4368-8BC0-F991E49798B1
20	Zlotkey	58B47DB1-E65C-439E-B7DE-C228B9955632

(1) Maximum search depth =trunc($\log_3$(number of rows))

(2) Index structure is small in size and is kept in memory, thus can be searched fast and easily.