INDEXES

INDEX IS ALSO ONE OBJECT. INDEX HELPS ME IN SEARCHING DATA FASE.INDEX IMPROVE PERFORMANCE OF SELECT QUERY. INDEX ALSO HAS SOME STORAGE. INDEX HAS ITS OWN PAGE. DO NOT CREATE INDEXES ON TABLES HAVING MIN. 1000 ROWS. NO NEED TO CREATE ANY INDEX ON THE TABLE HAVING MINIMIM 1000 ROWS. IT WILL GIVE U NEGATIVE PERFORMANCE. ON A TABLE WE CAN CREATE MAXIMUM 999 INDEXES. FOR SQL SERVER 2005 THE VALUE IS 249.

2 TYPES OF INDEXES

CLUSTERED INDEXES: IT WILL AGAIN RE ARRANGE THE WHOLE DATA IN THE SORTED ORDER IN HARD DISK. DATA IS ALWAYS PRESENT IN THE LEAF LEVEL.

NON CLUSTER INDEXE: LEAF NODE ALSO CONTAINS KEYS THAT STORES THE DATA OF THE ORIGINAL DATA. IT WILL NOT TOUCH THE DATA. IT WILL NOT PUT THE DATA IN SORTED ORDER. IT WILL STORE THE ADDRESS OF THE ALL THE RECORDS IN ANOTHER INDEX KEY CALLED INTERMEDIATE KEYS.

TERMINOLOGIES

HEAP: A TABLE WHICH HAS NO INDEX CREATED ON IT IS CALLED AS HEAP. FOR A HEAP INDER_ID WILL BE 0(ZERO) (QUEIRED FROM SYS.INDEXES).

WHEN SEARCH IS PERFORMED ON A HEAP , IT IS ALWAYS GOING TO BE A FULL TABLE SCAN. TABLE SCANS DANGEROUS FOR DQL SERVER PERFORMANCE (LARGE TABLE).

COMMAND TO FIND WHETHER TABLE IS HEAP OR NOT

SELECT *FROM SYS.INDEXES WHERE INDES_ID=0 AND OBJECT_ID('TABLE NAME')=OBJECT_ID

IF ANY INDEX_ID IS 0 IT IS CALLED HEAP, IF IT IS 1 IT IS CALLED CLUSTERED INDEX.

B-TREE

IT IS A DATA STRUCTURE WHICH HOLDS THE ORGANIZED DATA IN A ORGANIZED MANNER

INDEX KEY: INDEX KEY IS A KEY THAT CONTAINS THE LOCATION SPECIFIC OF THE DATA SELECTED IN THE TABLE. INDES KEYS PRESENT AT THE LEAF NODE MAP TO THE ACTUAL DATA IN A NON-CLUSTERED INDEX. BUT FOR A CLUSTERD INDEX DATA IS PRESENT AT THE LEAF NODE ITSELF HENCE INTERMEDIATE INDEX KEYS MAP TO THE LEAD NODE WHERE DATA IS STORED. THE SIZE OF THE INDEX KEY IS 900 BYTES FOR IMPLICIT KEY COLUMNS THE LIMIT IS 1700 BYTES. IF WE USE INCLUDE COLUMNS THEN WE CAN CREATE NON CLUSTERED INDEX EVEN INDEX KEY SIZE IS MORE THAN 900 BYTES. FROM SQL SERVER 2016 THE NON CLUSTERED INDEX KEY SIZE CAN NOT EXCEDD 1700 BYTES FOR IMPLICIT KEY COLUMNS THE KEY SIZE CAN NOT EXCEED 2600 BYTES.

---MAX COLUMNS IN A TABLE IS 1024

/*SMALL INTRO ABOT EXECTION PLAN

WHILE SELECT TABLE CHECK THE QUERY PLAN. U CAN SEE THE EXECUTOIN PLAN WITH CTRL+L SHORTCUT.

IF THE FLOW IS FROM RIGHT TO LEFT IT IS A DATA FLOW

IF THE FLOW IS FROM LEFT TO RIGHT IT IS A CONTROL FLOW(EXECUTION PLAN)*/

CLUSTERED INDEX
IN CLUSTERED INDES, DATA IS PRESENT AT THE LEAF NODE AND DATA IS PHYSICALLY ARREANGED AS PER THE SELECTED COLUMN. WE CAN CREATE ONLY ONE CLUSTERED INDEX FOR ANY VERSION OF SQL SERVER.. INDEX ID OF CLUSTERED INDEX IS 1.
EX:

CREATE CLUSTERED INDEX EMP_ENO_CI ON EMP(ENO)
WE CAN PUT UPTO 16 COLUMNS IN ONE CLUSTERD INDEX. PRIMARY CONDITION IS INDEX KEY SIZE CAN NOT MORE THAN 900BYTES.
ONCE WE CREATE CLUSTERED INDEX IT WILL RECREATE THE TABLE . CLUSTER INDEX IS NOTHING BUT TABLE. CLUSTERED INDEX FRESHLY CREATED THE WHOLE TABLE.
IF WE WANT TO MOVE THE TABLE FROM ONE FILE GROUP TO ANOTHER FILE GROUP LIKE FROM FG1 TO FG2, CREATE A CLUSTERED INDEX ON THAT TABLE IN FG2, SO THE TABLE WILL BE RECREATED IN FG2, THEN DROP THE INDEX. IN THIS WAY WE CAN MOVE THE TABLE FROM ONE FILE GROUP TO ANOTHER FILE GROUP. IF WE DROP INDEX IT WILL CONVERT INTO HEAP.
IN INDEX CREATIONS AND INDEX REORGANISATIONS TEMP DB PLAY CRUCIAL ROLE.
WE CAN NOT TAKE THE BACKUP OF TABLE, SO WE CAN NOT TAKE THE BACKUP OF INDESES ALSO. IF WE WANT BACKUP OF TABLE, PLACE THE TABLE IN TO ONE FILE GROUP AND TAKE THE BACKUP OF THAT FILE GROUP.

NON CLUSTERED INDEX.
IN NON CLUSTERED LEAF NODE POINTS TO THE ACTUAL DATA PAGES AND DATA IS NOT PHYSICALLY. WE CAN CREATE 249 (SQL 2005), 999(SQL 2008) NON XLUSTERED INDEXES.
WE CAN CRATE NON CLUSTERED INDEX ON A HEAP ALSO. WE CAN CREATE NON CLUSTERED INDEX ON A CLUSTERED INDEX. MEANS U R CREATED CLUSERED INDEX ON 5 COLUMNS , ANY OF THE 5 COLUMNS CAN BE A PART OF THE NON CLUSTERED INDEX.

DEMO
CREATE A TABLE
INSERT 4 RECORDS
CREATE CLUSTERED INDEX ON THE ABOVE TABLE
CREATE CLUSTERED INDEX CLUSTEREDINDEXNAME ON EMP(ENO) ON FG2
ANY HOW IT WILL DROP AND CREATE NEW TABLE ON FG2 CREATING CLESTER INDEX WILL TAKE DOWN TIME
CREATE CLUSTER INDEX AND CHECK , SELECT *FROM TABLENAME, CHECK WITH QUERY PLAN.

CREATE NON CLUSTERED INDEX ON THE SAME TABLE WITH ANTOEHR COLUMN.
CRATE NONCLUSTERED INDEX NONCLUSTERED INDEX NAME
ON EMP(ENAME) ON FG2
INDEX SEEK—SEEK WILL NOT DO FULL TABLE SCAN, IT WILL TOUCH THE EXACT COLUMNS , WILL SEARCH THE DATA BASED ON INDEX KEYS.
INDEX SCAN IS EQUAL TO TABLE SCAN.
COVERING INDEX: IF THE NON CLUSTERED INDEX IF WE COVER ALL THE COLUMNS THEN IT IS CALLED COVERING INDEX.
FILTERED INDEX:

WE CAN SEE ALL THE INDEXES ON A TABLE
SELECT *FROM SYS.INDEXES
IF CLUSTER ID IS 0 IT IS HEAP
IF IT IS 1, IT IS CLUSTERED INDEX
IF IT IS MORE THAN 1, THAT IS NON CLUSTERD INDEX.
--INSERT INTO EMP143 VALUES (1, REPLICATE('A', 1000), 10000)
REPLICATE FUNCTION WILL REPLICATE THE DATA THAT MANY TIMES, IN THE ABOVE EXAMPLE CHARACTER A WILL BE ADDED 1000 TIMES INTO THE TABLE.
*FROM 2012 WE HAVE ANOTHER FEATURE CALLED COLUMN STORE INDEX. IF WE USE THIS INDEX THE DATA WILL BE STORED IN COLUMNS MEANS IT WILL NOT TAKE ALL THE RECORDS IN THE COLOMNS, IT WILL CREATE COLUMS BASED ON OUR SEARCH. THIS INDEX IS USED FOR HISTORICAL DATA SO THIS IS READ ONLY INDEX. THIS INDEX WILL BE USED IN OLAP.

PAGE SPLIT

FRAGMENTATION
THE EMPTY SPACES THAT GET FORMED WHEN HUGE MODIFICATIONS OR DELETIONS HAPPEN ON A INDEX (INTERNALLY IN A TABLE).
INTERNAL FRAGMENTATION: EMPTY SPACES BETWEEN THE RECORDS. EXTERNAL FRAGMENTATION IS THE DATA MAY BE DELETED FROM THE OLD PLACE, OR MAY BE THAT COMMANDS WILL BE SIT SOME WHERE ELSE SO THE REGULAR PLACE THEY ARE USING ARE DELETED WILL CAUSE EXTERNAL FRAGMENTATION.
TO SEE THE FRAGMENTATION
DBCC SHOWCONTIG (TABLENAME, INDEX NAME)
PAGE DENSITY: HOW MUCH PERCENTAGE PAGE IS FILLED
SCAN DENSITY: HOW MANY PAGES ARE IN ORDER AND HOW MANY PAGES ARE NOT IN ORDER.

TO SEE THE PHYSICAL STATS BY USING DMV
SELECT *FROM SYS. DM_DB_INDEX_PHYSICAL_STATS
IF THE FRAGMENTATION PERCENT IS
1-10% NOT AN ISSUSE, WE CAN LEAVE

10-30% → REORGANIZE THE INDEX
30-100% → REBULID THE INDEX
ONLINE REBUID(SQL SERVER 2005) IS USED TO REBUILD THE INDEX WITH A NEGLIGIBLE DOWN TIME, IF WE WILL NOT USE ONLINE KEY WORD IT SHOULD TAKE MUCH DOWN TIME IF MY TABLE IS BIG IN SIZE. SO TO AVOID DOWN TIME WE WILL USE ONLINE REBUILDING. ONCE WE USE ONLINE REBUILD IT WILL USE THE OLD INDEX UNTILL THE COMPLETE REBUILD HAS DONE, ONCE REBUILD DONE IT WILL SWITCH WITH NEGLIGABLE DOWN TIME TO SWITCH.

COMMAND TO REORGANIZE INDEX
ALTER INDEX INDEXNAME ON FULLTABLENAME REORGANIZE

FOR SQL SERVER 2000
DBCC INDEXDEFRAG (DBNAME, 'TABLENAME', INDEXNAME);

COMMAND TO REBUILD THE INDEX
ALTER INDEX INDEXNAME ON FULLTABLENAME REBUILD

COMMAND TO REBUILD THE ONLINE INDEX
ALTER INDEX INDEXNAME ON FULLTABLENAME REBUILD (ONLINE=ON)

FILL FACTOR:WHEN WE CREATE OR MODIFY AN INDEX, THE PERCENTAGE OF FREE SPACE ALLOCATED IN THE LEAF LEVEL OF EACH INDEX PAGE DURING THE OPERATION.
LEAVING SOME SPACE IN THE PAGE FOR FUTURE USE.
R.CLICK ON THE INSTANCE, SELECT DATABASE SETTINGS, DEFAULT INDEX FILL FACTOR IF IT IS 0 THE PAGE WILL FILLED 100%, IT IS NOT A BEST PRACTICE TO KEEP THE DEFAULT INDEX FILL FACTOR AS 0, WE CAN FILL THE PAGE UP TO SOME PERCENTAGE.
PAD INDEX: SPACE RESERVATION AT INTERMEDIATE LEVEL. MEANS AT INDEX KEYS IT WILL RESERVE SOME SPACE FOR FUTURE USE. WE CAN NOT SET AN Y PERCENTAGE VALUES TO THE PAD INDEX. WE CAN ON OR OFF THE PAD INDEX. IF WE ON THE PAD INDEX IT WILL TAKE THE VALUE OF FILL FACTOR. MEANS IF MY FILL FACTOR IS 80%, IF WE ON THE PAD INDEX, SAME80% WILL BE APPLIED TO THE PAD INDEX.

DRFRAGMENTATION:REBUILDING OR REORGANIZING THE INDEX IS CALLED DEGRAGMENTATION.