



DBMS Implementation

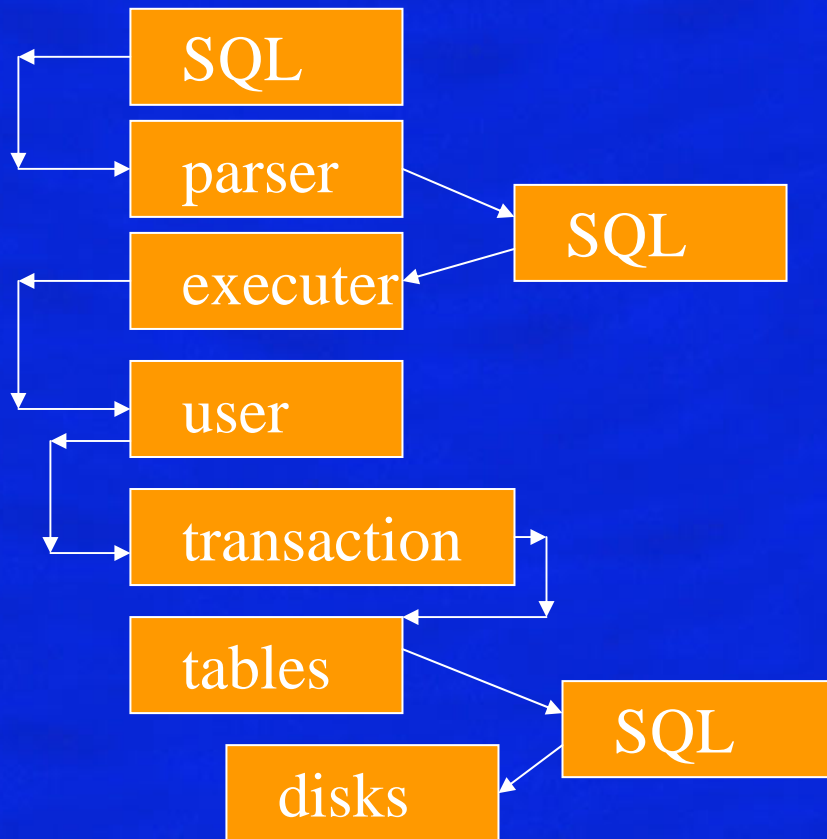
SET08104 Database Systems

Copyright @ Napier University



Implementing a DBMS

- The DBMS takes in SQL
- The SQL must be processed.
- The results of the SQL returned to the user/
Applied to the Database



Parser

- Must tokenise the SQL
- Time consuming
- Make use of an SQL cache.
- To increase usefulness, use placemarkers

```
SELECT empno FROM employee  
WHERE surname = ?
```

Executer

- Takes SQL tokens
- Converts tokens into Relational Algebra
- Optimises the RA
- Sends RA requests down the tree
- Repackages results into Tables

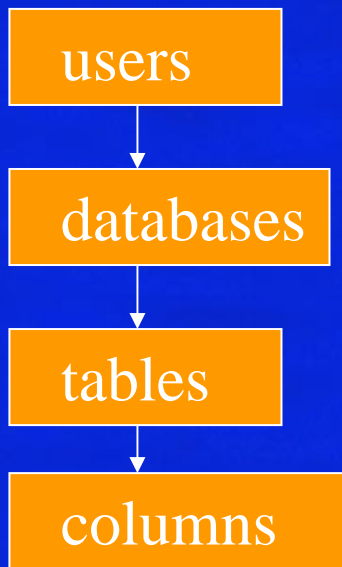


- Users – user level security
- Transactions – run queries independently
- Tables – all underlying concepts are table based.
- Cache – cache disk traffic as much as possible
- Disks – the only area of persistent storage

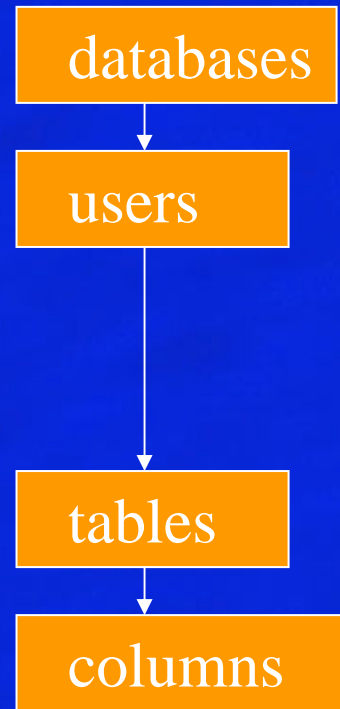


User Concept

MySQL



Oracle



Tablespaces

- Tablespace/database . Table . Column
- Accessing the table car, column owner, in tablespace vehicles would be:

```
SELECT vehicles.car.owner  
FROM vehicles.car
```


Disk vs Memory

- Speed – Memory 1000 times faster
- Size – Disk 100 times bigger for same cost
- Persistence – Disk remembers through loss of power
- Access Time – ns vs ms
- Block size – Disk is 4 KB blocks, memory “word”.

Disk Arrangements

- Need to find what we want on disk without loading more than we need.
- Need to handle disks effectively, especially with:
 - Indexes
 - Transaction logs
 - Disk Requests
 - Data Prediction

Indexing

- Want a catalogue or index whereby we can find things quickly without looking at each block.
- Many different techniques.
- Here consider only two:
 - Hash tables
 - Binary Trees



Hash Tables

- Hash tables are a fast way to find things:
 - They have a hash function
 - They have buckets
- The hash works out which bucket to stick information into...

Hash Tables

- Consider

```
CREATE TABLE test (  
    id            INTEGER primary key  
    , name        varchar(100),)
```

```
Insert into test values (1,'Gordon');
```

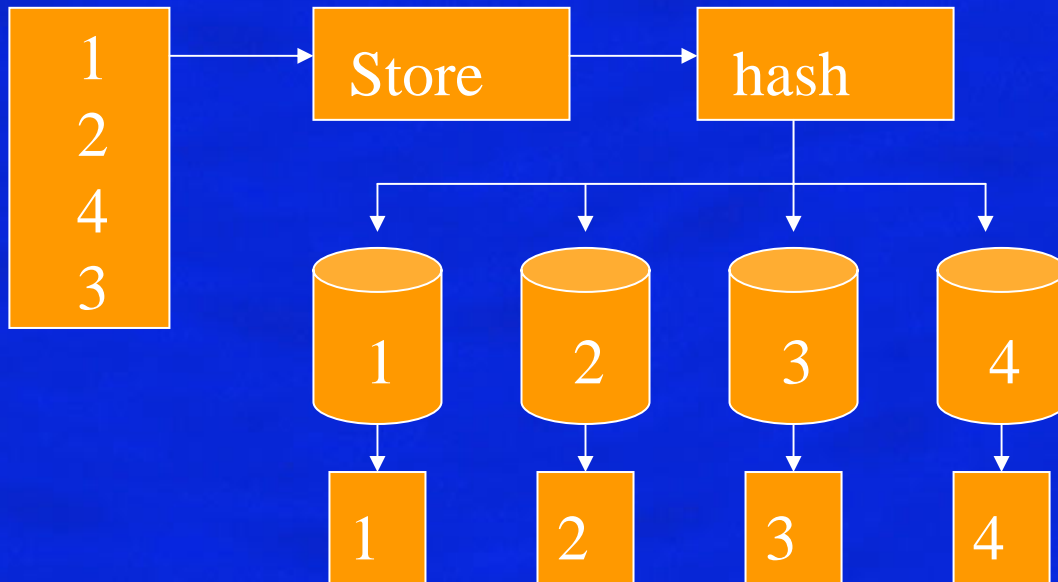
```
Insert into test values (2,'Jim');
```

```
Insert into test values (4,'Andrew');
```

```
Insert into test values (3,'John');
```

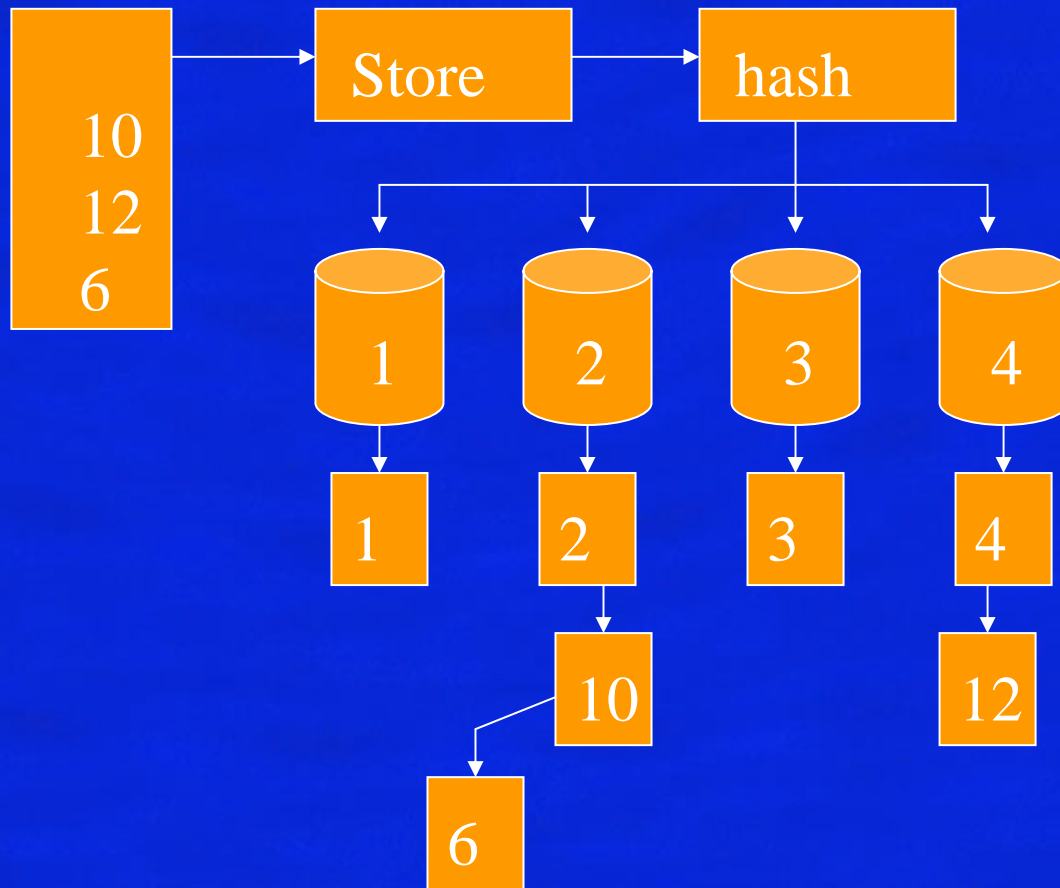


Initial Example





Adding 10,12,6



Binary Trees

- Modern Databases rely on binary trees for indexing
- One of the most modern version of a binary tree is called a B+ Tree.
- The B+ means that the depth of the tree remains roughly in balance.

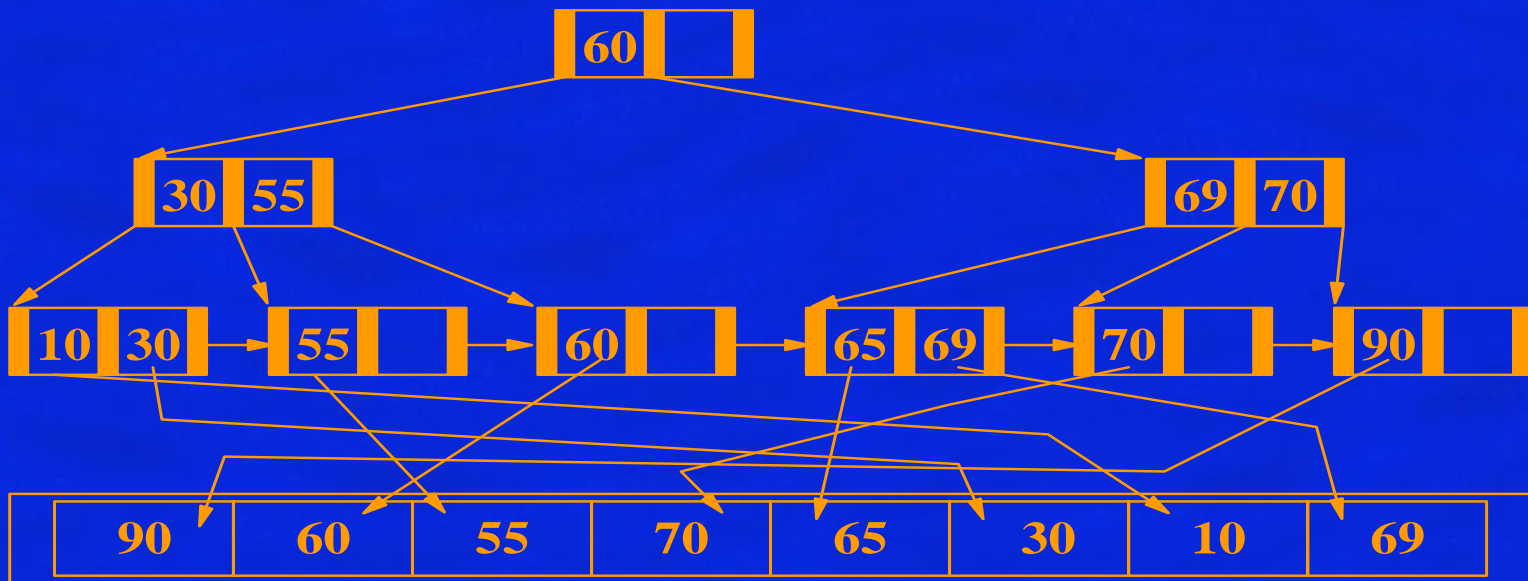
B+ Tree Index

With B+ tree, a full index is maintained, allowing the ordering of the records in the file to be independent of the index. This allows multiple B+ tree indices to be kept for the same set of data records.

- the lowest level in the index has one entry for each data record.
- the index is created dynamically as data is added to the file.
- as data is added the index is expanded such that each record requires the same number of index levels to reach it (thus the tree stays ‘balanced’).
- the records can be accessed via an index or in insertion order.



B+ Tree Example



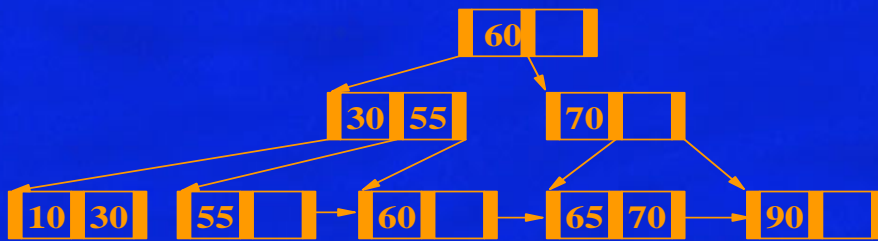


B+ Tree Build Example

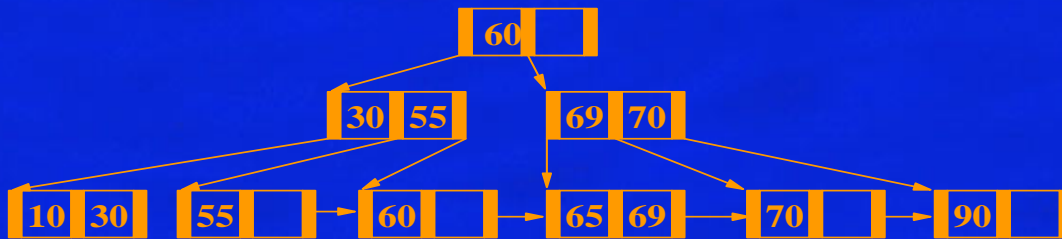




B+ Tree Build Example Cont...



Add 10



Add 69

Index Structure and Access

- The top level of an index is usually held in memory. It is read once from disk at the start of queries.
- Each index entry points to either another level of the index, a data record, or a block of data records.
- The top level of the index is searched to find the range within which the desired record lies.
- The appropriate part of the next level is read into memory from disc and searched.
- This continues until the required data is found.
- The use of indices reduce the amount of file which has to be searched.

Costing Index and File Access

- The major cost of accessing an index is associated with reading in each of the intermediate levels of the index from a disk (milliseconds).
- Searching the index once it is in memory is comparatively inexpensive (microseconds).
- The major cost of accessing data records involves waiting for the media to recover the required blocks (milliseconds).
- Some indexes mix the index blocks with the data blocks, which means that disk accesses can be saved because the final level of the index is read into memory with the associated data records.

Use of Indexes

- A DBMS may use different file organisations for its own purposes.
- A DBMS user is generally given little choice of file type.
- A B+ Tree is likely to be used wherever an index is needed.
- Indexes are generated:
 - (Probably) for fields specified with 'PRIMARY KEY' or 'UNIQUE' constraints in a CREATE TABLE statement.
 - For fields specified in SQL statements such as CREATE [UNIQUE] INDEX indexname ON tablename (col [,col]...);
- Primary Indexes have unique keys.
- Secondary Indexes may have duplicates.



Use of Indexes cont...

- An index on a column which is used in an SQL 'WHERE' predicate is likely to speed up an enquiry.
 - this is particularly so when '=' is involved (equijoin)
 - no improvement will occur with 'IS [NOT] NULL' statements
 - an index is best used on a column which widely varying data.
 - indexing and column of Y/N values might slow down enquiries.
 - an index on telephone numbers might be very good but an index on area code might be a poor performer.



Use of Indexes cont...

- Multicolumn index can be used, and the column which has the biggest range of values or is the most frequently accessed should be listed first.
- Avoid indexing small relations, frequently updated columns, or those with long strings.

Use of indexes cont...

- There may be several indexes on each table. Note that partial indexing normally supports only one index per table.
- Reading or updating a particular record should be fast.
- Inserting records should be reasonably fast. However, each index has to be updated too, so increasing the indexes makes this slower.
- Deletion may be slow.
 - particularly when indexes have to be updated.
 - deletion may be fast if records are simply flagged as ‘deleted’.

Shadow Paging

- The method for handling transaction logs discussed to far is relatively simplistic.
- Modern databases usually use Shadow Paging.
- The main difference is that the log has copies of whole disk blocks, rather than changed attributes.


Algorithm

1. If the disk block to be changed has been copied to the log, jump to 3.
 2. Copy the disk block to the transaction log.
 3. Write the change to the original log.
- On commit, delete the log copies
 - On abort, copy the blocks back to their original place.



Disk Parallelism

- Oracle
 - 2.8M control01.ctl
 - 2.8M control01.ctl
 - 2.8M control01.ctl
 - 11M redo01.log
 - 11M redo01.log
 - 11M redo01.log
 - 351M sysaux01.dbf
 - 451M system01.dbf
 - 3.1M temp01.dbf
 - 61M undotbs01.dbf
 - 38M Users01.dbf

- 
- DBMS designed for multiple users.
 - One user running 1 query may be as fast as 100 similar difficulty queries running simultaneously.
 - Bottom line: efficient databases are ones kept full of queries...