**ADM notes**

**Chapter 1 – Information Models**

Data – Unprocessed figures

Information – Processed data, meaningful and useful (Answer who, what, where, when)

Knowledge – Gain from information, use info to make decision (Answer how)

Online transaction processing (OLTP) – captures, stores and processes data from transactions in real time.

Online analytical processing (OLAP) – use complex queries to analyze aggregated historical data.

Categorization of information

* Source
  + Primary info, original source
  + Secondary info, second-handed info
  + Internal info, within company
  + External info, outside company
* Nature
  + Qualitative, descriptive data
  + Quantitative, measurement data
* Level
  + Strategic info
    - Top-level management
    - for long-term decisions
    - (etc: find new market, plan business growth)
  + Tactical info
    - Middle-level management
    - tactical planning and decision-making
    - (etc: choose supplier, forecast sales, prepare budget)
  + Operational info
    - Lower-level management
    - operating planning
    - (etc: schedule shipping, monitor stock)
* Time
  + Historical, data stored in a period of time
  + Present, data created in the current work window
  + Future, data created to predict the future

Information capturing and representation (page 22)

Metadata – data about data, information about another set of data, example: library catalog, data dictionary

Queries:

* Procedural queries
  + Created by programmers using sequences of instructions
  + Requires technical knowledge
* Declarative queries
  + Only state WHAT you need instead of HOW
  + Example: SQL
* Navigational queries
  + Searchers know where they want to go to find something
  + Word (HP) or link ([www.hp.com](http://www.hp.com))

Information security:

* Confidentiality, authorization and authentication when accessing data
* Integrity, protection from improper data modification
* Availability, ensure timely and reliable access to data

Threats of information security (page 33)

Information security examples (page 34)

Information assurance – measures to protect CIA of information

Non-repudiation – assurance that someone cannot deny something

Database audit trace (page 35)

**Chapter 2 – Database systems**

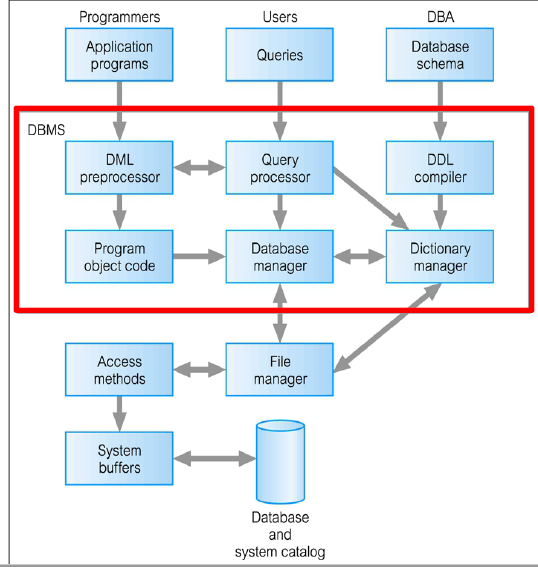
Problems with data dependency (page 5, 6)

* Waste of space
* Hard to maintain
* Data will be inconsistent
* No data integrity

Database management system (DBMS) – allows users to define, create, maintain and control access to the database.

Functions:

* Store, update and retrieve data
* Concurrency control services, allow multiple users to make changes on database simultaneously.
* Solutions:
  + Locking & 2-phase locking
  + Time-stamping
  + Versioning
* Backup and recovery services
* Security - encryption, authentication authorization, views
* Integrity services, avoid incorrect data with rules and constraints
  + Types of integrity constraints (page 13)
    - NOT NULL
    - Unique
    - Primary key
    - Foreign key
    - CHECK
* Transaction support, all updates are made or none of them is made
* Data dictionary management, system catalogue / metadata
  + Stores information (page 16)



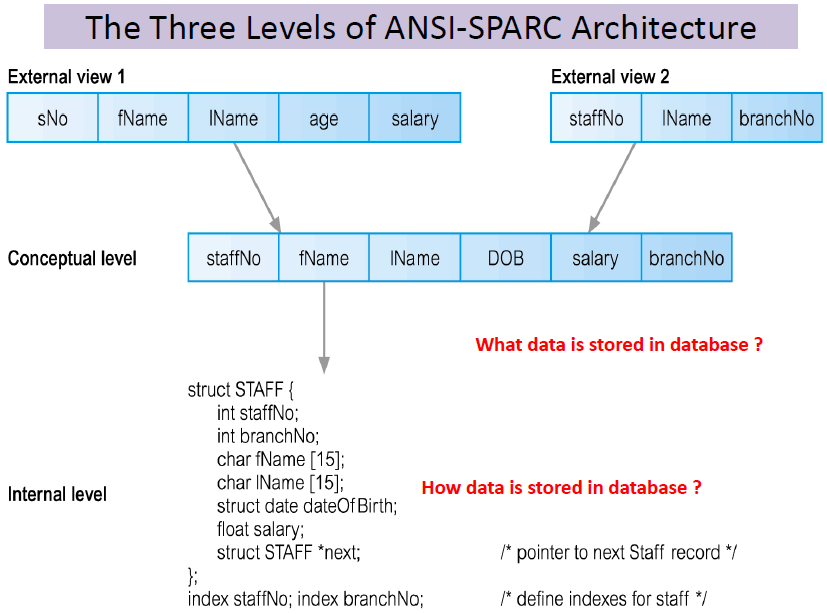
Components of DBMS (page 20)

* Query processor, transform queries into low-level instruction for database manager
* Database manager (page 23)
  + Authorization control
  + Command processor
  + Integrity checker
  + Query optimizer, optimize query execution
  + Transaction manager
  + Scheduler, ensures concurrent operations
  + Recovery manager
  + Buffer manager, transfer data from main memory to secondary storage
* DML preprocessor, converts DML statements into function calls in host language
* DDL compiler, converts DDL statements into set of tables containing metadata, then stored in system catalog
* Dictionary manager/catalog manager, manages system catalogs

ANSI-SPARC Architecture – American National Standards Institute, Standards Planning and Requirements Committee, design standard for a DBMS

Three-level architecture

* External, users’ view of the database, show what’s relevant on websites.
* Conceptual, community view of the database, shows **what** data is stored and the relationship of the data.
* Internal, physical representation of the database, **how** data is stored



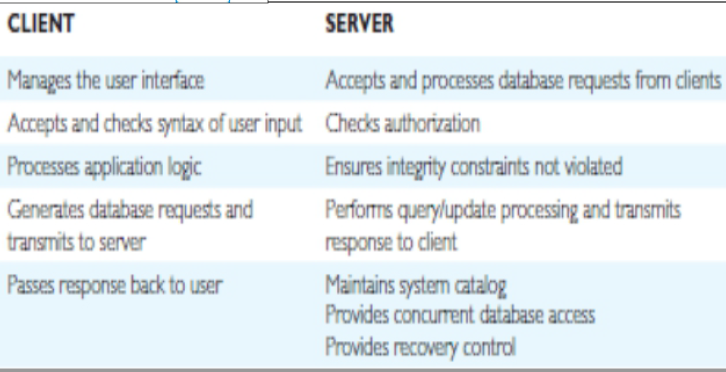
Objectives of Three-level architecture (page 27, 28)

Data independence (page 29, 30)

* Logical data independence
* Physical data independence

Multi-user DBMS architecture

* Teleprocessing
  + One computer with single CPU and a number of terminals
  + Processing performed within the same physical computer, terminals are “dumb”, connected through cables
* File-server
  + Connected to several workstations across a network
  + Database links file-servers, workstations run DDMS and applications
  + Disadvantages (page 34)
    - Significant network traffic
    - Redundant DBMS on each workstation
    - More complex control on concurrency, recovery and integrity
* Client-server
  + Two-tier
  + Client (tier-1) – runs applications and manages UI
  + Server (tier-2) – holds database and DBMS.
  + Summary



* + Three-tier
    - Advantages (page 38)
      * Thin client, client requires less expensive hardware
      * Centralized application maintenance
      * Easy tier replacement
      * Separate business logic and database functions, easy to implement load balancing.
      * Maps naturally to web environment
    - Transaction processing monitors
      * Control data transfer between clients and servers to provide a consistent environment
  + N-tier
    - Provides more flexibility and scalability
    - API is hosted by applications servers

Distributed DBMSs (page 42)

**Chapter 3 – Data modeling**

Importance of data modeling (page 4)

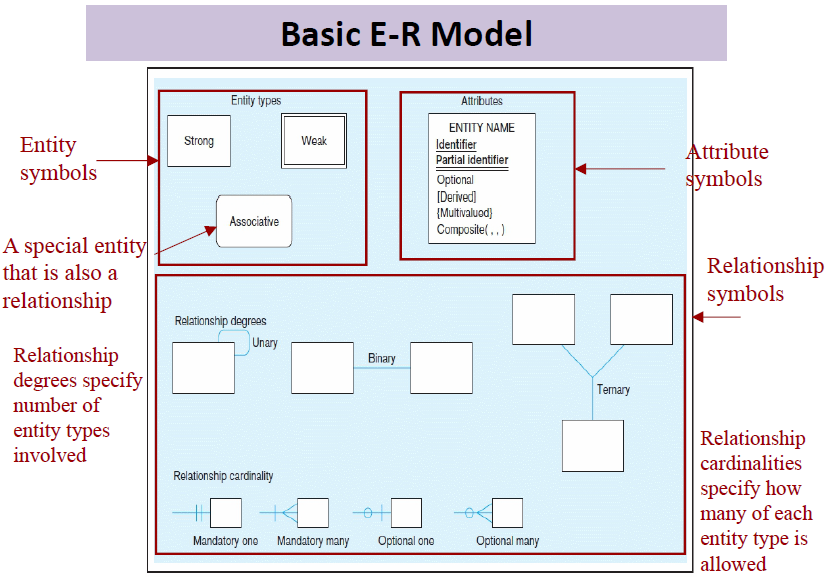
* Facilitate interaction among designer, application programmer and end-users

Data modeling describes

* Structural part
* Manipulative part
* Set of integrity rules

Entity-relationship model (page 10)

Basic E-R model symbols



Early data models

* Hierarchical model (page 12)
* Network model (page 13)
* Limitations:
  + Data stored in rigid, predetermined relationships
  + No DDL existed, difficult to change data structure
  + No simple query language

Relational model

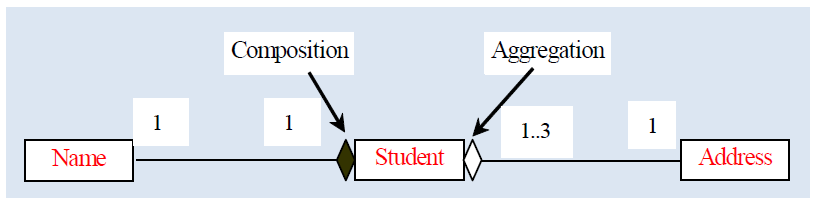
* Represented by a collection of tables
* Each table has columns called attributes
* Degree – number of attributes
* Cardinality – number of tuples/rows in a relation

Integrity constraints (page 16)

* Entity integrity, don’t allow multiple rows with same identity, primary key
* Domain integrity, restricts data types, or data range
* Referential integrity, requires data at another table exists, foreign key

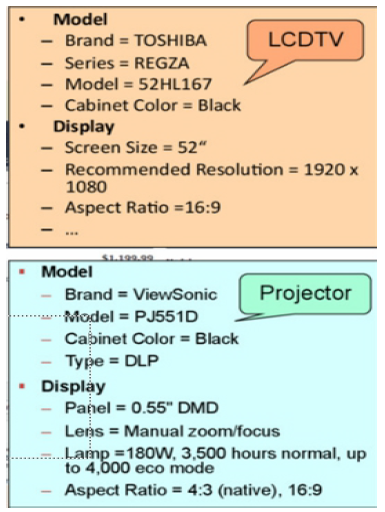
Object-oriented model (page 17)

* Abstraction of real-world entity
  + Attributes/state – properties of an object
  + Behavior/operation – how object reacts and acts
* Objects shares similar characteristics are grouped in classes
* Inheritance (page 18)
* Aggregation and composition (page 19)
  + Aggregation – not dependent
  + Composition – dependent
  + Example

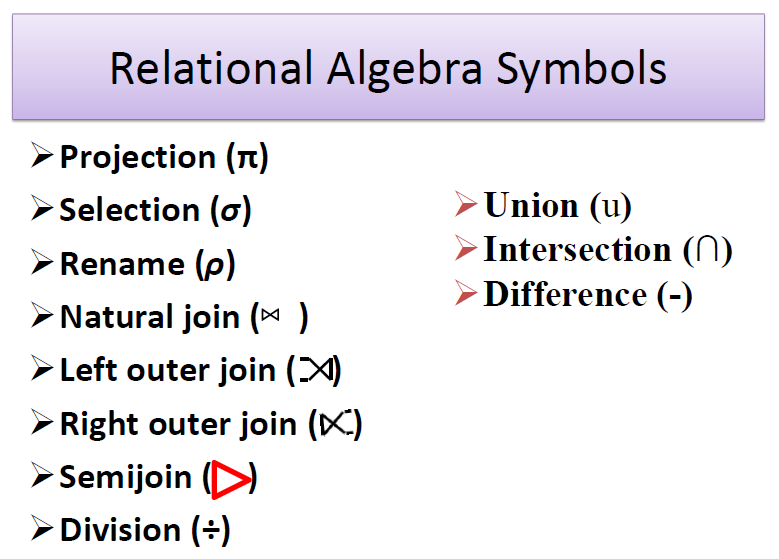


* Represented using UML
* Why (page 21)
  + Conventional data models are inadequate
    - Can’t model complex or unstructured data
    - Can’t model processes / dynamic behavior
    - Can’t be reused
  + OO model is natural in representing the real world

Semi-structured data model

* Structured data, follows a strict format
* Semi-structured data, may not be rigid, regular or complete, example scientific data
  + 
  + Forms: JSON, XML, emails
  + Advantages:
    - Schema can be easily changed
    - Data exchange between different types of databases
    - Used for data sources that cannot be constrained by schema
  + Compare with relational data (page 30)
* Unstructured data, limited indication of data type, example text file storing images, sound, videos

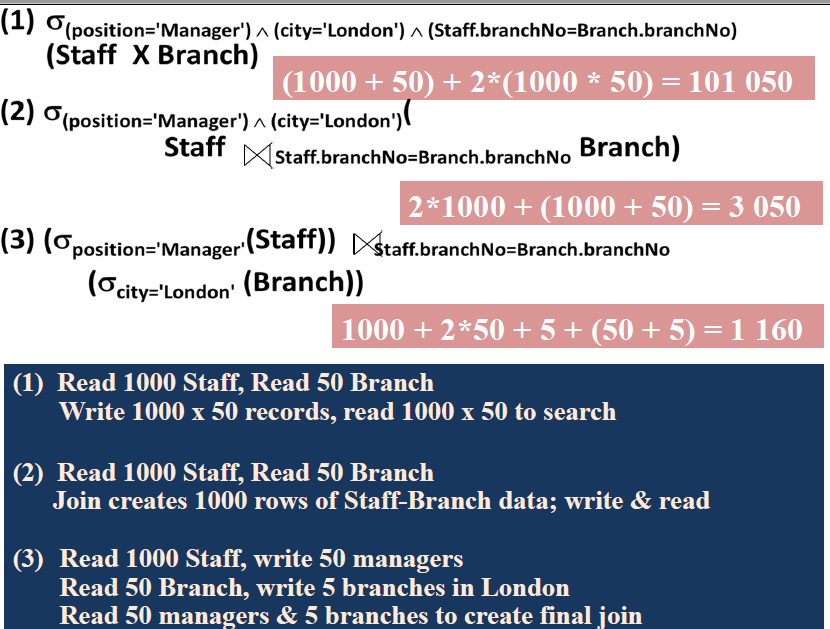
**Chapter 4 – Relational model**



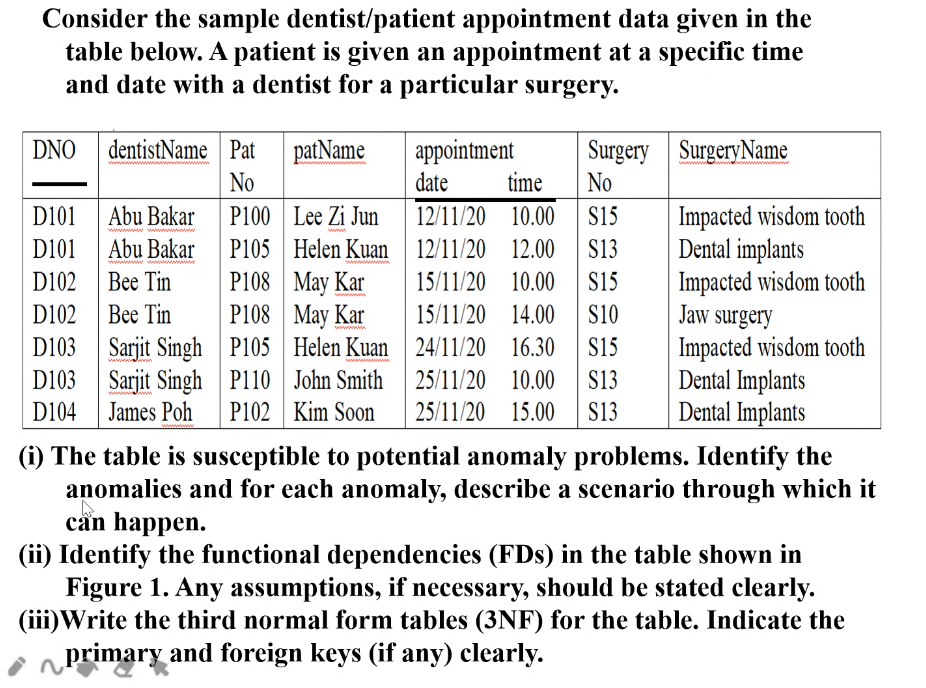
Basic steps in query processing

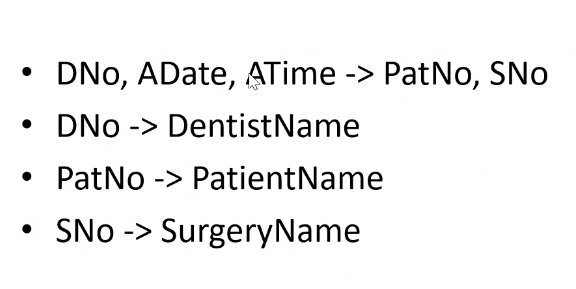
1. Parsing and translation
2. Optimization
   1. Chooses efficient execution plan
   2. Minimize resource usage, reduce total execution time
3. Evaluation (page 56)
   1. Strategy 1: Materialization (page 57)
      1. Evaluate one operation at a time, starting from the lowest level
      2. Use intermediate results to evaluate then next level
      3. Time consuming and expensive to generate and store temporary files
   2. Strategy 2: Pipelining (page 58)
      1. Evaluate several operations simultaneously, pass results to the next operation
      2. Avoid constructing temporary results
      3. AKA stream-based, on-the-fly processsing

Example for query Optimization (page 55)



**Chapter 5 – Relational database design**



ii) 

SID, CourseCode → Grade, CGPA

SID → SName, Program

CourseCode → CourseTitle

**Chapter 6 – Indexing**

Create index

Syntax:

CREATE [UNIQUE] INDEX indexName ON tableName (columnName [ASC|DESC])

Example:

CREATE INDEX state\_index ON Customer (cus\_state)

Drop index

DROP INDEX indexName

Index

* Use search key k to find the desired index entry k\*
* The index entry k\* contains info to locate one or more data records

Steps:

1. Enter search key k
2. Find the index entry for k
3. Once index entry is found, the row can be directly accessed

Why not indexing all columns?

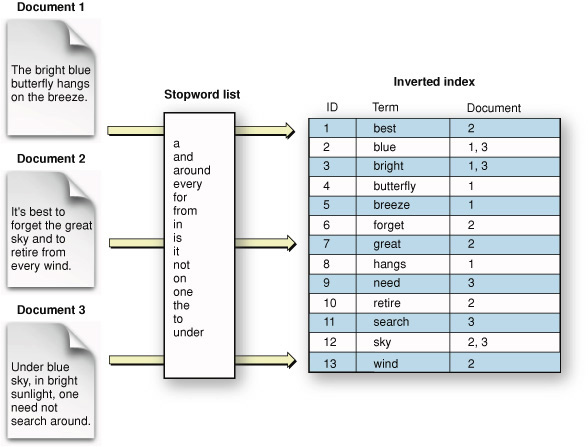
* Not practical
  + DBMS needs even more processing power for index-maintenance processing, especially for tables with
    - Many attributes
    - Many rows
    - Many inserts, updates or deletes

Indexing structure

* Hash indexes
* B-tree indexes
* Bitmap indexes

Major text indexing methods

* Inverted index
  + For very large collections of documents, use in most search engines
  + Composed of:
    - Vocabulary
    - Occurrences
  + Advantages:
    - Allows fast full text searches
    - Easy to develop
    - Most popular
  + Disadvantages
    - Large storage overhead and high maintenance costs on database operations.
  + Example:



* Signature files
  + Usually not for large database sizes
* Suffix trees and arrays

**Chapter 7 - Physical database design**

Issues of physical database design

* Storage media
* File organization
* Indexes

File organization

* Heap
* Sequential
* Hash

Indexing structure

* Hash index – good for simple and fast lookup operations
* B-tree index – Most common type of index used in databases
* Bitmap index – used in data warehouse applications
* Join index – used in data warehouse applications

Variable-length records

* Arise in several ways
  + One or more fields have variable length
  + One or more fields are repeating
  + One or more fields are optional
  + File contains records of different types

File organization

* Heap (unordered) files
  + Records are placed on disk in no order
  + New files are inserted at the end of the file
* Sequential (ordered) files
  + Records are ordered by the value of a specific field
* Hash files
  + Records are placed according to hash functions

Index evaluation metrics

* Access time
  + For equality searches (specific value) and range searches (specific range of values)
* Insertion time
  + Time to find the correct place, insert the new data and update the index structure
* Deletion time
  + Time to find the item to delete, delete the item and update the index structure
* Space overhead
  + Space to store the index structure

Classification of indexes

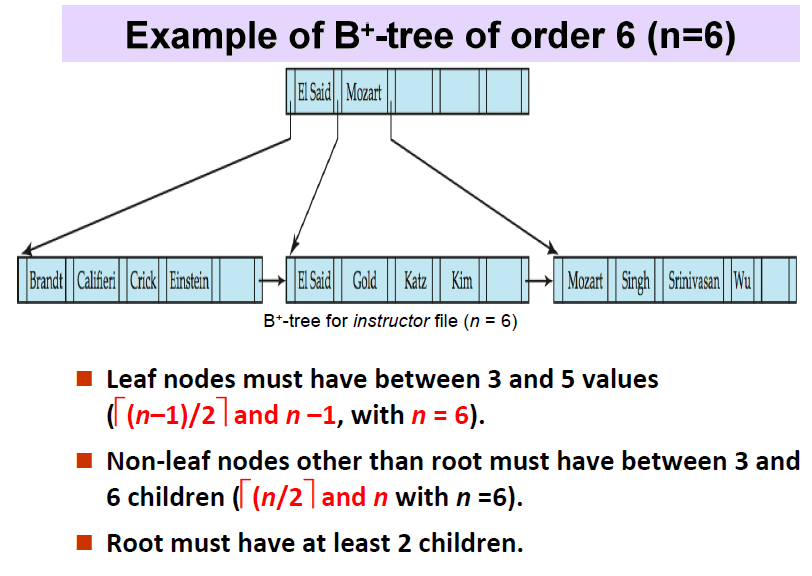
* Clustering vs non-clustering
  + Clustering:
    - ordering of index matches the ordering of value
    - index that defines the physical order the records are stored in the database
  + Non-clustering:
    - Like book index
    - Stored separately, may have multiple non-clustered indexes for each table
    - Slow search, search for index using the search-key value, then use the index to locate the record from the table
    - Fast insert and update, order of record will only be updated in the index table
  + <https://www.spotlightcloud.io/blog/when-to-use-clustered-or-non-clustered-indexes-in-sql-server>
  + <https://www.geeksforgeeks.org/difference-between-clustered-and-non-clustered-index/>
* Dense vs sparse index
  + Dense index: each value of search key matches the index
    - Faster when locating records
    - No sorting is required, all search keys have an index
  + Sparse index: index records only have some search key
    - Less space
    - Less maintenance overhead
    - Least search-key value in the block
    - Sorting is required to put the data into blocks
  + <https://yetanotherdevblog.com/dense-vs-sparse-indexes/>

Multilevel index

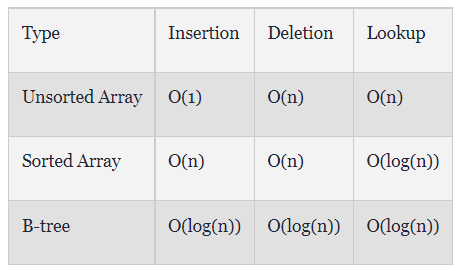
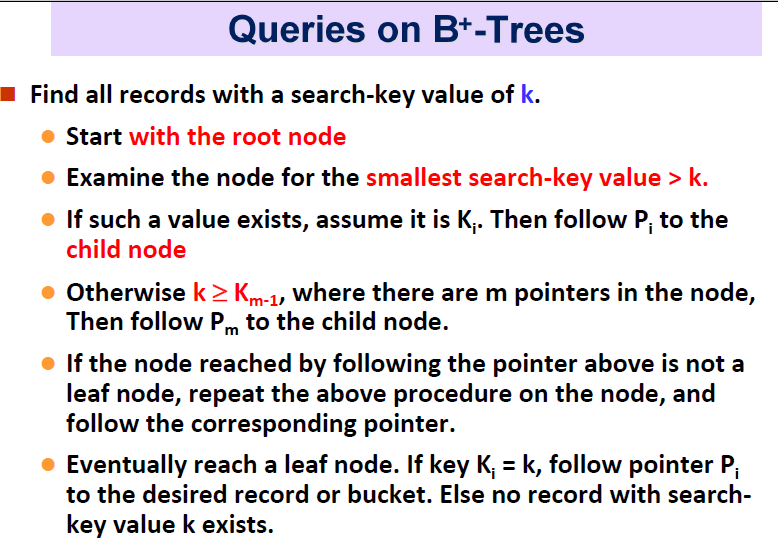
* When index don’t fit in the memory, construct multilevel index to reduce disk accesses
* Two level of index
  + Outer index – sparse index on main index
  + Inner index – the main index file
* Indices at all level must be updated on insertion or deletion from the file

B+- Tree index files

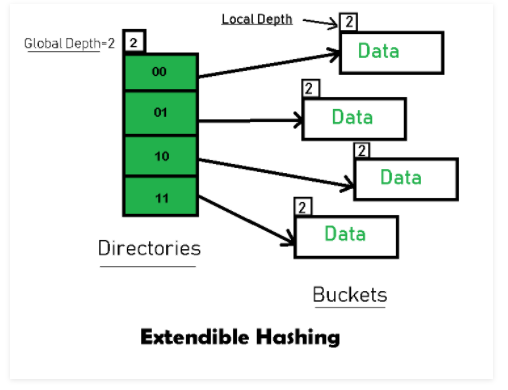
* Characteristics
  + All paths from root to leaf are in the same length
  + N = max no of pointers per node
  + A leaf node has between (n-1)/2 and n-1 values
  + Example



* Need to be balanced, all leaf nodes are at the same level to prevent excessive block access during tree search
* Searching, inserting, deleting (<https://www.javatpoint.com/dbms-b-plus-tree> )



Hash index

* Organize the search keys with their associated record pointers, into a hash file structure
* Static hashing
  + Maps search-key values to a fixed set of bucket addresses
  + Bucket overflow might occur
    - Insufficient buckets
    - Skew in record distributions
      * Multiple records have the same search-key value
      * Hash function produces non-uniform distribution of key values
  + Solve by:
    - Using overflow buckets, which are buckets chained together in a linked list
  + Cons of static hashing
    - Performance degraded when database expands
    - For anticipated （预料到的）growth, a lot of spaces are wasted at start
    - When database shrinks, more space are wasted
* Dynamic hashing
  + Allow hash function to be modified dynamically
  + Forms:
    - Extendable hashing
      * 32-bits
      * Bucket address table size = 2^i , 0 <= i <= 32
      * Value of i grows and shrinks based on the database
      * Then, size will be smaller due to merging and splitting of buckets
      * 
      * <https://www.geeksforgeeks.org/extendible-hashing-dynamic-approach-to-dbms/>
      * Ad:
        + Data retrieval is much more efficient, no overflow occurs
        + No data loss occur as the storage capacity increases dynamically
      * Dis:
        + Directory size will increase significantly when several records are inserted into the same directory in non-uniform distribution.
        + Fixed bucket size
        + Waste on memory when difference between global depth and local depth is really big
        + Complicated

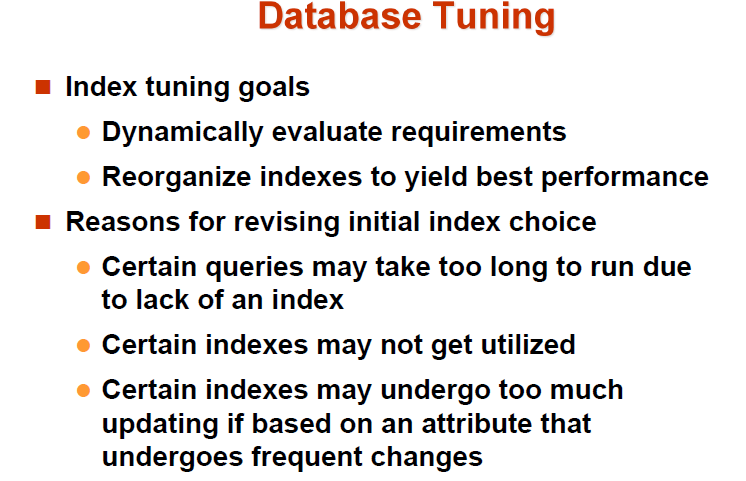
Rules for using indexes

* Use on larger tables
* Index the primary key
* Index frequent search fields
* Index fields in SQL ORDER By and GROUP BY commands
* Avoid using index for fields with long values before compressing it
* DBMS has limits on number of indexes per table
* Be careful of indexing attributes with null value, DBMS might not recognize it

Comparison of the indexing techniques

* The cost of routine re-organization of indexes
* The frequency of insertions and deletions
* Optimizing average access time at the cost of worst-case access time
* Hashing is good for searching in specific value, while B tree is good for searching a range of values

Database tuning



**Chapter 8 – Transaction processing**

Views

* Subset of database
* User can view but not make changes on the original tables

Integrity control

* ensure data is accurate, valid and consistent
* Entity integrity
  + No same identity, primary key
* Domian integrity
  + Restricting data, AS Decimal
* Referential integrity
  + Ensure existence of one row for another row, foreign key

Authorization

* Restrict data access and operations on data
* Oracle privileges are granted to users in matrix of:
  + Subject
  + Object
  + Action
  + Constraints

Authentication schemes

* Password
* Two factor authentication
* Three factor authentication
* Biometric authentication – fingerprints, retina scans
* Third-party mediated authentication – using secret keys, digital certs

Transaction support

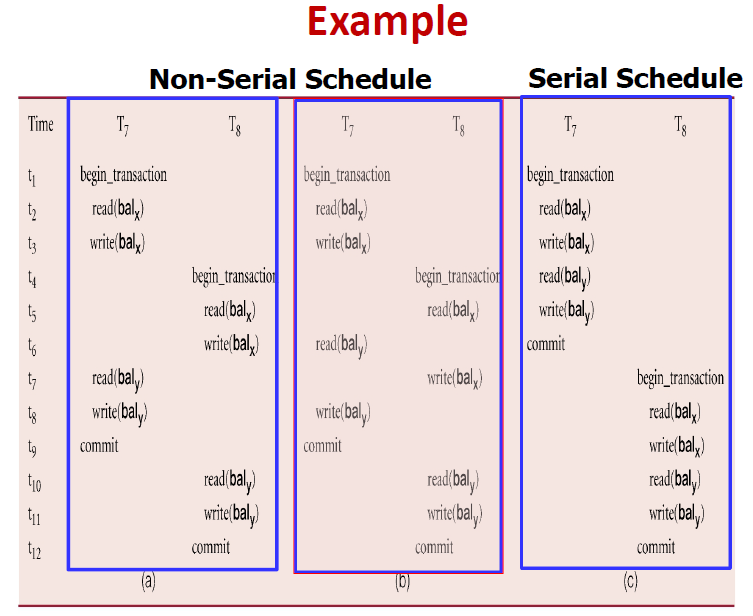
* Actions made on the data in database (SELECT, INSERT, UPDATE and DELETE)
* Boundaries:
  + BEGIN TRANSACTION
  + COMMIT, transaction is completed successfully, change to another state
  + ROLLBACK, transaction is aborted, back to previous state
* Transaction ACID properties
  + Atomicity, all or nothing
  + Consistency, database is consistent
  + Isolation, partial effects or incomplete transaction should not be shown in other transactions
  + Durability, effects of committed transaction are permanent

Concurrency control

* Managing simultaneous operations without them interfere with each other
* Potential problems caused by concurrency:
  + Lost update problem
    - One success update is overridden by another success update
    - Only allow others to read after the current update is success
  + Uncommitted dependency
    - Transaction can see intermediate results before it has committed
    - Only allow others to read after the current update is committed or aborted
  + Inconsistent analysis problem
    - One transaction reads several values but other transactions is still performing updates
    - Only allow others to read those values after the current update is completed

Serializability

* Scheduling the transactions
* Serial schedule – without any interleaved operations
* Non-serial schedule – with interleaved operations
* Serializable - finding non-serial schedules that are equivalent to some serial schedule
* Ordering of read/writes is important:
  + Only read, do not conflict so order is not important
  + Read and write different items, do not conflict so order is not important
  + Read and writes same item, order is important



Concurrency control techniques

* Conservative approaches
  + Locking
    - Widely used
    - Types:
      * Shared (read) lock
        + Transaction that can read but not update it
        + Many users can hold shared lock on the same item
      * Exclusive (write) lock
        + Transaction can read and update the item
        + Only one user can hold exclusive lock of an item
    - Level (Page 33)
      * Database
      * Table
      * Page/block
      * Record
      * Attribute
    - Two-phase locking (2PL)
      * Growing / expanding phase – acquires locks without releasing any locks
      * Shrinking phase – releases locks without acquiring new locks
    - Deadlock
      * Each transaction is waiting for locks held by another transaction to be released
    - Deadlock handling
      * Timeouts
        + Requests only wait for a system-defined period of time, transaction is aborted and restarted automatically
      * Deadlock prevention
        + Prevent deadlock in the first place

Wait-die

Transaction with smaller timestamp (older) waits, aborted otherwise

Wound-die

Transaction with smaller timestamp (older) preempts other transaction to abort, wait otherwise

* + - * Deadlock deletion and recovery
        + Detects and breaks deadlock
        + Wait-for-graph (WFG)

Create node for each transaction

Draw arrows for transaction who waiting for locks held by another transaction

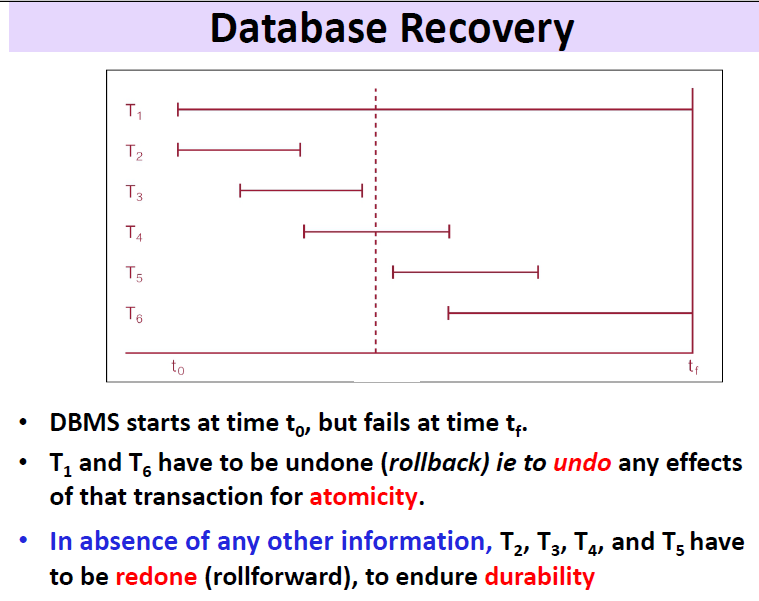
Deadlock exists when cycle is found

Created at regular intervals

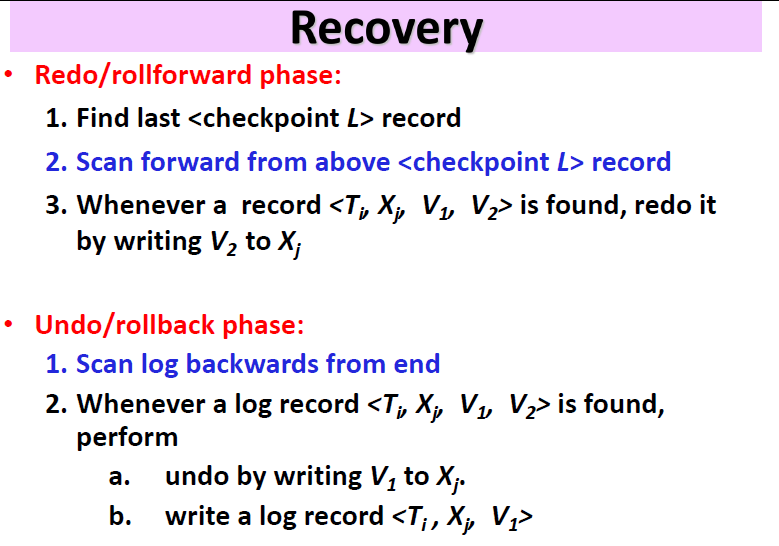
* + Timestamping
* Optimistic for rare conflicts
  + Versioning
    - Optimistic approach
    - Increase concurrency
    - Assume concurrent update is rare
    - Check conflict at commit, if conflict occurs, transaction is rolled back and restarted
    - Three phases:
      * Read
        + Read data in local copy
      * Validation
        + Read-only transaction, checks that the data read is still same as current values
        + Update transaction, checks if it leaves a consistent state
        + If not, transaction is aborted and restarted
      * Write
        + When validation is success, updates made to the local copy are applied to the database
    - Ad:
      * Deadlock free
      * Maximum parallelism
    - Dis:
      * Rerun transaction when aborted
      * High loads when conflict occurs

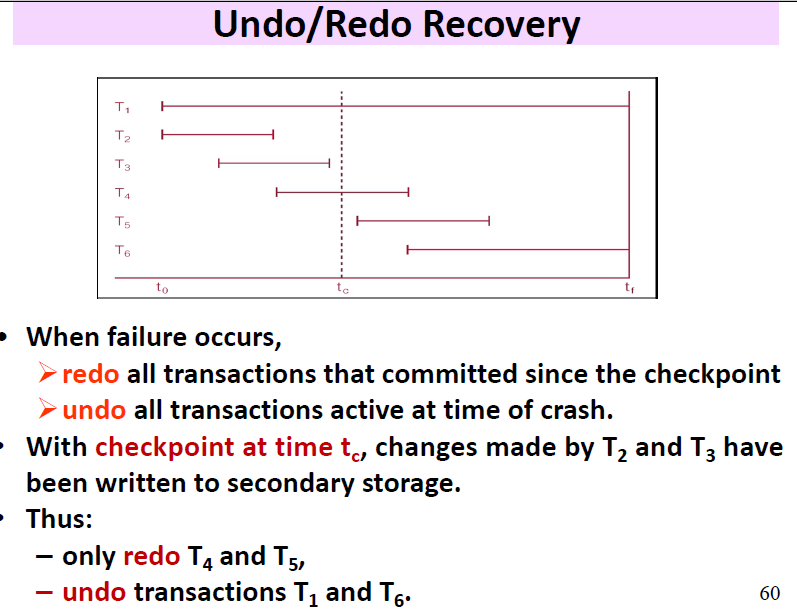
Database recovery

* Types of failure
  + System crashes
  + Media failures
  + Application software errors
  + Natural physical disasters
  + Carelessness or unintentional destruction of data or facilities
  + Sabotage



* Recovery facilities (pg 52)
  + Backup mechanism
  + Logging facilities
    - Logging files contains:
      * Transaction identifier
      * Type of log record (start, insert, update, delete, abort, commit)
      * Data item identifier affected
      * Before-image of data item
      * After-image of data item
  + Checkpoint facility
    - Checkpoint – point of synchronization between database and log file, buffers are force-written to secondary storage (pg 54)
  + Recovery manager
    - Module that restores the database to a correct condition when failure occurs, then resumes the processing
    - Procedures (pg 55)
      * Disk mirroring – switch between identical copies of databases
      * Restore/rerurn – reprocess transactions
      * Backward recovery – apply before-images
      * Forward recovery – apply after-images





Database failure responses (pg 61, 62)

* Aborted transactions
  + Causes – human error, invalid data, deadlock, hardware failure
  + Recovery – rollback
  + Alternatives – rollforward to state right before the abort
* Incorrect but valid data
  + Causes – carelessness or unintentional human mistakes
  + Recovery – rollback if found soon enough
  + Alternatives – correct the error manually, reprocess transactions from checkpoints before the error occurred
* System failure
  + Causes – power loss, operator error, communication loss, software failure
  + Recovery – switch to duplicate database
  + Alternative – rollback, restart from checkpoint
* Database destruction
  + Causes – disk drive failure
  + Recovery – switch to duplicate database
  + Alternative – rollforward, reprocess transactions

**Chapter 9 – Distributed database**

|  |  |  |
| --- | --- | --- |
|  | Distributed database | Centralized database |
| Definition | Collection of shared data physically distributed over the network | Database that is located, stored and maintained in a single position |
| Advantages | Increase reliability/availability   * One gone but others are still there   Local control over data   * Each site has own control on database data, security, logging, and etc.   Modular growth   * Easier to add in computer and data   Lower communication cost, faster responses for some queries   * Data center is located near to the users |  |
| Disadvantages | Software cost and complexity  Processing overhead   * Need proper coordination among data at different sites   Affect data integrity   * Harder to maintain data integrity between different sites   Slower responses for certain queries   * Get data from data centers which are far from users |  |

Types of distributed databases (DDBMS)

|  |  |
| --- | --- |
| Homogeneous DDBMS | Heterogeneous DDBMS |
| * All sites run same DBMS product * Allow mixture of different OS * Easier to design and maintain * Good expandability | * Sites run different DBMS products with different data models * Sites implement their own database and consider to integrate with other databases later * Translations are required for smooth integrations * Using gateways, convert language/model of different DBMS into language/model of the relational system |