CS 5012: Foundations of Computer Science	UNIVERSITY OF VIRGINIA DATA SCIENCE INSTITUTE

Exam 2 Study Guide

Topics
Data Structures – Trees
Abstract Data Types – Priority Queues
Databases – Relational Algebra
Databases – Entity Relationship Diagrams (ER)

The readings provided below are in addition to class material and in-class exercises *Note: The topic of SQL will not be on this exam, instead, it will be on the final exam.*

Readings

- MSD textbook (on Collab)
  - o Ch. 6 (especially: 6.1, 6.2, and 6.4) [Data Structures / ADT]
  - o Ch. 7 (not 7.5) [Trees / Binary Trees / BST / Heaps]
- Introduction to Algorithms (3rd edition) by Cormen et al.
  - o Ch. 6 (6.5) [Priority Queues]
- Database System Concepts (6th edition) by Silberschatz, Korth, and Sudarshan
  - o Ch. 2 [Relational Model]
  - o Ch. 6 [Formal Relational Query Languages / Relational Algebra (RA)]
  - o Ch. 7 [Entity Relationship Diagrams (ER)]
- *Additional textbook*: Discrete Mathematics and its Applications (7th edition) by Rosen (See Resources on Collab)
  - o Ch. 13 (especially section on regular expressions)

Data Structures: Trees

- General definition and terminology
  - Recursive definition
- Tree Traversals
  - What is it
  - Why might this operation be useful/needed
  - The three common tree traversals for binary trees
    - Pre-order traversal
      - Prefix expression
    - In-order traversal
      - Sorts values from smallest to largest
      - Infix expression
    - Post-order traversal
      - Depth-first search
      - Postfix expression
    - Understand these traversal methods are applied recursively
  - Given a tree, know how to perform each of these three traversals
- Traversal applications
- Binary Search Trees (BST)
  - Binary search tree property
  - Finding and inserting in BST
  - How to traverse a BST so that the nodes are visited in sorted order?
  - Deleting from a BST – not covered

Abstract Data Types – Priority Queues

- A priority queue (PQ) is an Abstract Data Type (ADT)
  - An abstract data type (ADT) is a computational model for data structures that have similarity in behavior
  - An abstract data type is defined indirectly, only by the operations that may be performed on it and by mathematical constraints on the effects of those operations for performance efficiency
  - ADTs are purely theoretical entities used to
    - Simplify the description of abstract algorithms
    - Classify and evaluate data structures

- o An ADT may be implemented by specific data structures
- o An “abstraction” to hide implementing details (e.g. microwave oven) ●

The most important element deserving priority can be Min or Max depending on the nature of application

● In a PQ, an element with “high priority” is served before an element with “low priority” ● If two elements have the same priority, they are served according to their order in the queue

● A queue has ‘FIFO’ structure

● (Whereas a stack has ‘LIFO’ structure)

● Reasons one might jump a (standard) queue – why break the normal priority of a queue o Operating system scheduling
  - ▪ FCFS (*non pre-emptive*)
  - ▪ SJF (*non pre-emptive*)
  - ▪ SRTF (pre-emptive version of SJF)

● Priority Queue Operations

● Priority Queue – several ways to implement it
- o Heap data structure is the preferred option
- o (vs. array implementation)
  - ▪ Sorted or unordered array
  - ▪ Insertion and deletion
- o (vs. list-based implementation)
  - ▪ Sorted or unordered list
  - ▪ Insertion and deletion

● Binary Heap structure
- o Special version of a binary tree with special structure and heap-order properties

● Tree Data Structure
- o Basic vocabulary and tree elements
- o Trees are also used for sorting and searching different types of data. The insert/ delete operations can be done faster in tress than in arrays
- o Arrays are generally fixed size unless re-dimentioned at runtime. A tree naturally grows to hold an arbitrary, unlimited number of objects to go with the needs of the user

● Binary heap
- o Balanced binary tree
- o Heap must be a complete tree
- o Arrays are generally fixed size unless re-dimentioned at runtime
- o A tree naturally grows to hold an arbitrary, unlimited number of objects to go with the needs of the user
- o A binary heap is a heap data structure created using a binary

tree o It can be seen as a binary tree with two additional constraints:

    ▪ Shape property
    ▪ Order (heap) property

o Minheap vs. maxheap

    ▪ Minheap used
    ▪ A 'min' heap example will be used where the key with the lowest value is at the root

o Binary heap inserting and retrieving the smallest value

o Heaps as 1-D arrays

    ▪ Level order
    ▪ Heapform

o Inserting a node into a Heap (minheap)

o Deleting a node from a Heap – successor nodes (and where to find them!)

Database Systems – Relational Algebra (RA)

- Sailors and Bank Examples – Relational Algebra (RA) 1

RA:

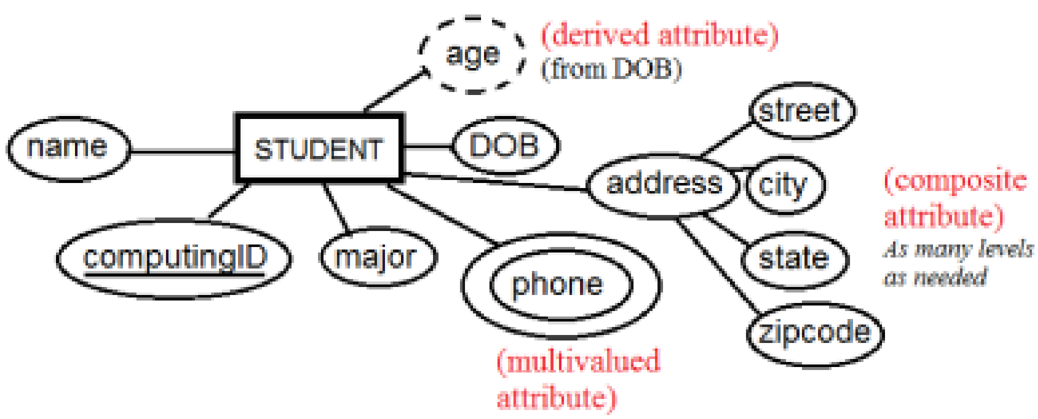
- Select
- Project
- Select & Project (combined)
- Union
- Difference
- Cross (*Cartesian product*)
- Rename
- Both *write* and *interpret* queries
- The Outer Join – *including the null matches*
- Aggregate Functions - Group by ● Division
- Combinations of the above…

Database Systems – Entity-Relationship (ER) Diagrams

- Entity-Relationship Model is used to design a database
- Consists of entities and relationships
- Entity: Any object that is distinguishable from any other
    - Have attributes
    - Entity Set is a set of entities of the same type
    - Value Domain of Attributes: set of permitted values for each attribute
    - Simple or composite: char, int, etc. or name with first, middle, last ○ Single values or Multiple Values: multiple phone numbers for a person ○ Derived: (Age can be found from DOB and today's date)
    - Null applies here too
- Relationship: an association between 2 or more entities

    - Mathematical
    - Two important properties: participation and cardinality
    - Participation: who is involved?
    - Cardinality: how are they involved together?
    - Total Participation: every entity in the set participates in at least one relation, e.g. Every loan MUST have a borrower – double edge
    - Partial Participation: can have entities not participating – single edge
    - Cardinality: 1 to 1, 1 to many, many to 1, many to many
    - Relationships can have their own attributes: Descriptive Attr. deposit-date
    - Recursive Relationship – e.g. Works-For with people
- Basic E-R Diagram Drawing
    - Rectangle – Entity Sets
    - Diamond – Relationship Sets
    - Lines – Cardinality
    - Ellipses – Attributes
    - Double Ellipse – Multi-valued
    - Dashed Ellipse – Derived
    - Arrow – "one" relationship
    - Undirected – "many" relationship

Recall the following diagrams:



Relationship "**sit-in**" between entities "**Student**" and "**Chair**".
"Only one person will sit in one chair" (*Cardinality*: one-to-one)

- What makes a good entity set?
- What makes a good relationship?
- Design Decisions

- Entity sets vs. attributes – person & telephone #
- Rule of thumb – if you need more info about it, make it an entity – if it is the info, make it an attribute
- Entity sets vs. Relationship sets
- Binary vs. n-ary Relationship sets
- Strong vs. Weak entity sets

- Reduction of an E-R Schema to Tables
  - Primary keys allow entity sets and relationship sets to be expressed uniformly as tables which represent the contents of the database.
  - A database which conforms to an E-R diagram can be represented by a collection of tables.
  - For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set.
  - Each table has a number of columns (generally corresponding to attributes), which have unique names.
  - **Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram**.
  - Schema statements:
    - **`TableName (`**`attr1, attr2, attr3, attr4`**`)`**
    - **`TableName (`**`attr1 int, attr2 varchar, attr3 date`**`)`**
  - See: Relational Schema for Bank Enterprise.pdf