CS143: Database Systems
Homework #5

1. Suppose you have 2 relations, R(A, B, C) and S(B, C, D, E).

   You have a clustering unique (no duplicate keys) B+-tree index on attribute A for relation R. Assume this index is kept entirely in memory (i.e., you do not need to read it from disk).

   For relation S, you have a non-clustering non-unique B+-tree index for attribute B. Furthermore, assume that the index is kept in memory (i.e. you do not need to read it from disk).

   Other relevant data:

   - 500 tuples of R are stored per block on disk.
   - $|R|$ = 750,000 (number of tuples of R).
   - 100 tuples of S are stored per block on disk.
   - $|S|$ = 250,000 (number of tuples of S).
   - For every R tuple, there are roughly 5 tuples in S with R.B = S.B.

   750,000/500 = 1,500 blocks to read R
   750,000 * 5 = 3,750,000 blocks of S read
   3,750,000 + 1,500 = 3,751,500
   750,000 * (5,000/100) = 37,500,000

   You want to execute the following query:

   ```
   SELECT * FROM R, S WHERE R.B=S.B AND R.C=S.C
   ```

   We present you with the following query plans:

   ```
   For every block Bi of R, retrieved using the clustered index on A for R
       For every tuple r of Bi
           Use the index on B for S to retrieve all of the tuples s of S such that s.B=r.B
               For each of these tuples s, if s.C=r.C, output r.A, r.B, r.C,
                                                              s.B, s.C, s.D, s.E
   ```

   How many disk I/Os are needed to execute this query plan? 3,751,500

   Now assume that we have a clustering non-unique B+-tree index for attribute C of S. For every R tuple, there are roughly 5000 tuples in S with R.C = S.C. Assume that all of the tuples of S that agree on attribute C are stored in sequentially adjacent blocks on disk (that is, if more than one block is needed to store all of the tuples with some value of C, then these blocks will be sequentially located on the disk). The index is kept in main memory.

   Using this new index we came up with a second query plan:

   ```
   For every block Bi of R, retrieved using the clustered index on A for R
       For every tuple r of Bi
           Use the index on C for S to retrieve all of the tuples s of S such that s.C=r.C
           For each of these tuples s, if s.B=r.B, output r.A, r.B, r.C, s.B, s.C, s.D, s.E
   ```

   Now analyze each of the two plans more carefully in terms of their behavior regarding accesses to disk. Your analysis should consider the behavior of the number of I/Os and access time. Explain which of the two plans is therefore better under what circumstances. For the analysis of access time, you do not need to compute a concrete number. Just include in your analysis what accesses to disk are sequential accesses and which ones are random accesses and discuss its consequence on overall access time.

Both plans require the same 1,500 I/O operations to read the blocks of R.
The first plan requires an additional (750,000 * 5) random I/Os.
The new plan requires an additional 37,500,000 sequential I/Os.
Thus the new plan requires approximately 10 times more I/O (5 vs. 50 blocks of S per tuple in R). If the cost of random access is more than 10 times the cost of sequential access, then the second plan will require less time to complete.

2. You are to design a database that maintains information for producing a weekly television guide for a given region (such as Northern California). The data should include information about television shows, television networks, cities, channels, show times, etc. For starters, you may make the following assumptions:

   - A given channel in a given city is associated with one network.
   - A given show is either owned by a network (and shown on a channel associated with that network) or is a local show and may be shown on any channel.
   - Not all shows are shown in all cities, and the days and times for a given show may differ from city to city.
   - You may ignore cable channels, which generally are not city-dependent.

   Please feel free to make additional assumptions about the real world in your design, as long as the assumptions are reasonably realistic and are stated clearly as part of your solution.

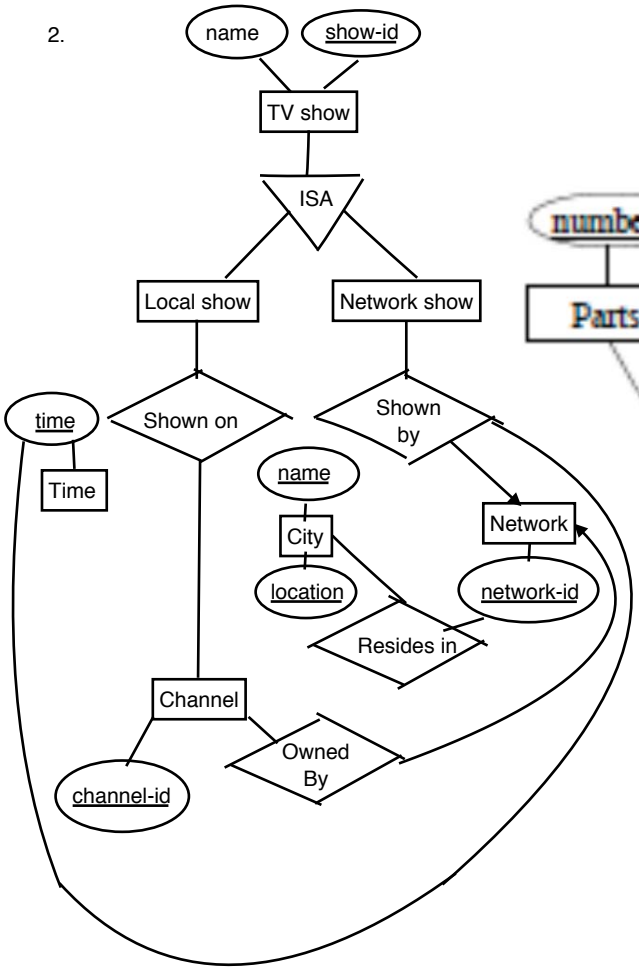
   Specify an entity-relationship diagram for your database. Don't forget to underline key attributes and include arrowheads and double lines.

   Note that this question is fairly open-ended and there is no single right answer, but some designs are better than others.

3. This problem is based on an E/R design for a database used in a manufacturing company shown in Figure 1. This database stores information about parts. Each part has a part number, which uniquely identifies the part. A part may in fact be an assembly, which consists of some number of one or more subparts. For example, a bicycle might be described as an assembly consisting of one frame and two wheels; a frame is just a basic part; a wheel is an assembly consisting of one tire, one rim, and 48 spokes. Each assembly is also associated with the cost of assembling its subparts.

   Convert the E/R diagram to relations. For the translation of subclasses, assume that we generate multiple tables for specialization and that a subclass does not inherit non-key attributes from its superclass.

2.



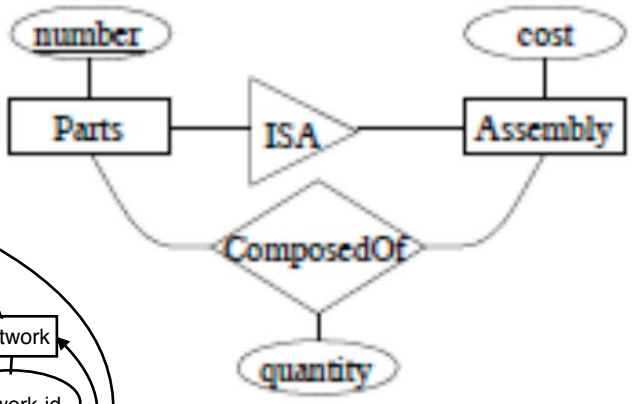


Figure 1: E/R Diagram for Question 3

Parts(number)
Assembly(number, cost)
ComposedOf(assemblynumber, partnumber, quantity)