Section A

1-

1. NOT RELEVANT ANYMORE - DBMS are able to model this data accurately and ensure consistent representation of this data, while at same time abstract the information from various part of the business.
2. NOT RELEVANT ANYMORE - no they should consider scaling out, as just increase the number of computers in their cluster will enable them to fit the demand as much as they want, if they no longer require those process power they can rent them out.
3. deliveryAddress is a multivalued attribute and is a partial key. Part is a weak entity that relies on a parent to uniquely identify it
4. not sure what expression tree is, but I think they meant rational algebra:  
   π(Emp.name, Dept.deptName)(σ(Dept.deptId = Emp.dept, Dept.budget > 3)(Emp X Dept))

Section B

2-

1. Transaction: is an action or sequence of action that transform the database from one state to another, it has four properties known as the ACID properties:  
   **Atomic**: A transaction must complete all its work or no partial work was ever done. AKA all or nothing contract.  
   **Consistent**: A transaction must transform the database from a consistent state to another.  
   **Isolated**: The partial effects of incomplete transaction shouldn’t be visible to other transactions  
   **Durable**: The effect of committed transactions are permanent and must not be lost due to later failure.
2. PL/SQL introduced a programmable procedures into SQL. with PL/SQL we are able to encapsulate more logic into the declarative code of SQL.   
   It has the advantage of bringing variables, constant declaration, exception handling, control structure and modularization
3. Index is a data structure that allows DBMS to locate records quickly and thus increasing the speed. The DS is a key value pair of the search query and its location in the data file
4. Primary index: spare index which is defined on order data file based on key field.  
   Clustering index: Sparse index which is defined order data file based on non-key field (clustering field) so more than one record can correspond to single value  
   Secondary index: a dense index which is defined on a non-ordering field of the data field. can have as many as needed secondary indices   
     
   N.B sparse index: index on some of the search value  
    Dense index: index on all search values
5. Log file contains information about all updates to database: transaction records and checkpoint records.   
   It is used for rolling back the database to unaltered state, it is used in auditing and many other places.

Section C

3.

1. [picture](https://drive.google.com/open?id=0BxSfLz74ghGpbThCTC1BRFAyNkU), ~~note I opted out of linking employee to building directly as one can derive it from dept id, but I am not sure if I should include it, I would write exactly this in the exam.~~ NVM the second part of the question makes it obvious you do need a relation called located\_at which is total from the employee (1:m) and leading to a building (1:1)
2. SELECT average(emp.salary) as AverageSalary  
   FROM Employee emp  
   NATURE JOIN located\_at loc   
   GROUP BY loc.building\_address
3. SELECT dept.name   
   FROM Department dept  
   NATURAL JOIN located loc  
   NATURAL JOIN Building build   
   WHERE build.address = ‘25 Oxford Road’ OR build.address = ‘10 Market Street’
4. It should be DxB as a building could span many department, and many department can span many building.

4)

1. It creates anomalies, in case an employer change their name, or a project change its name will create also contains a lot of redundant data   
   can be broken into a 3nf for easily to avoid that
2. Academic (idNumber, title, name)  
   AcadPhon(academic\_number [fk:Academic.idNumber], number)  
   HeadOfSchool(academic\_id[fk:Academic.idNumber], secretaryName)  
   School(schoolId, name)  
   WorksIn(academic\_id[fk: Academic.idNumber], school\_id[fk: School.schoolId], startDate)  
   HeadOf(head\_id[fk: HeadOfSchool.academic\_id], school\_id[School.schoolId])  
   Manages(head\_id[fk: HeadOfSchool.academic\_id], academic\_id[fk: Academic.idNumber])
3. A → E (transitivity from A → B, B → E)   
   AD → EF (composition)  
   A → BC (union)
4. I would use hash when what I want to look for is exactly what I am searching for it is quite fast and efficient for lookups indexing is useful when we want to pattern match or we want the approximate location of a file and indexing can provide constant access to a start of file allowing easier bulk update hash reduce the size of the lookup key but not sure how is this for an answer