COIT20247 - Database Design & Development
Tutorial – Physical design

Note: If you are in a scheduled tutorial, it would be wise to attempt the practical questions first while your tutor is present.

Theory questions

Answer each of the questions below. You should be able to answer most of these questions within about 3 to 5 sentences. Refer to the lecture slides to find the important points about each of these questions.

1. Describe what is meant by **physical design**. Where does it fit in the overall database development process?

2. What are the two competing goals to physical design? Describe at least one physical design circumstance where these two goals compete directly.

3. Define what is meant by **denormalisation.**

4. Why should you be careful using denormalisation?

5. Define what is meant by **horizontal partitioning**. Describe a situation where horizontal partitioning might be useful.

6. Define what is meant by **vertical partitioning**. Describe a situation where vertical partitioning might be useful.

7. What is the difference between a **unique** and **non-unique index**?

8. Would you define a unique index on a *Student name* field? Why or why not? Would you define a non-unique index on a *Student name* field? Why or why not?

9. List the seven rules of thumb for selecting an index.

Practical questions

10. If you have access to a computer, download the *PhysicalDesignTute.mdb* database file from the course website. This database contains four tables. Each table has a number of attributes. However, no physical design choices or data integrity measures have been implemented for these tables. All attributes have been defined with the default data type of text, which is inappropriate. Modify the design of this database as follows:

    a. Choose appropriate data types and sizes for each attribute.
    b. Set primary keys as appropriate
    c. Add validation rules where appropriate.
    d. Decide which attributes should never be null and set their *required* property to true.
    e. Add default values where appropriate
    f. Add unique indexes for any attribute that should not contain duplicate values.
    g. Create relationships between the tables as follows (make sure you enforce referential integrity):
        i. Customer.CustomerID and Order.CustomerID

ii. Order.OrderID and Orderline.OrderID
iii. Orderline.PartID and Part.PartID

Assume that:

- All customers are Australian; limit post codes and states accordingly.
- Part name should never be duplicated.
- The stock-keeping practices in this organisation do not permit negative quantities in stock or negative quantities ordered.
- Quantities in stock will typically be around a few thousand units each.
- Quantities in stock may be whole units only.
- All IDs are to be numerical.
- It is expected that the Customer and Order tables may grow to hundreds of thousands of rows.
- The Part table is expected to grow to a few thousand rows.

Note:

- Although physical design does not normally include actual implementation of tables, working with real tables for this tutorial question will help ground your understanding of this process.
- It is expected that you will need some assistance from your tutor to find your way around Microsoft Access.

11. Consider the following list of attributes. What data types, sizes, range controls, integrity rules, null/not null options and indexes would you specify for them? Assume that storage efficiency is of ***no concern***, that is, you don't have to choose data types that will save space. Your only concerns are a) being able to store the valid range of data values, and b) **data integrity.** Justify your answers. Assume you are using the data types and data integrity facilities available in Microsoft Access.

    a. A *Quantity* field, such as for the number of parts ordered.
    b. A *Student number* field – such as for a CQUniversity student
    c. A *Post code* field, assuming that only Australian post codes are permitted
    d. A *student name*, keeping in mind that many international names are very, very long
    e. A *height* field, assuming that the users want to keep very precise measurements.
    f. An *address* field. Keep in mind that some addresses can be five lines long. Would you break this down into simpler attributes?
    g. A *Price* field, for example, *price per kilogram* when purchasing coffee.
    h. A *Date of birth* field. Should future dates of birth be possible?

12. Now assume that you have to be careful about storage space. What data types or sizes etc would you change for the previous question?

13. Did you specify range controls for the *Quantity*, *Height* and *Price* fields in the previous question? If so, what did you specify? Did you know that some coffees can cost $400/kg? Did your range control allow for this?

14. Did you specify a maximum length for the names and address fields in the previous question? What happens if the user has to enter a name or address longer than you've allowed for? (When storage space is a concern, these choices *are not easy*. There isn't really a "right" answer for these questions, but you need to be aware of the consequences of your choices.)