DATABASE AND INFORMATION RETRIEVAL

DR. ELIEL KEELSON

LECTURE 05 – PHYSICAL DATABASE DESIGN

Physical Database Design

- A Database Management System, usually abbreviated to DBMS, is the software system that provides the facilities necessary to design and support a database application.

- A DBMS typically has many components some of which are described below:
  - Database Engine
  - Query processor
  - Schema manager
  - Forms manager
  - Report generator
  - Data dictionary

Features of a DBMS

- The **database engine** is the part of the DBMS that does the actual work of storing and accessing application oriented data (i.e. tables, rows etc.) from physical storage.

The functions performed by the engine are indicated below:

1. Physical data management and accessing including index management. Indexes are special tables created and maintained by the engine which speed up retrieval of data from the database.

2. View management. A view is essentially a ‘virtual table’ generated from a query on normal tables.

Features of a DBMS

3. Accessing of Data Dictionary.
4. Transaction control.
5. Security: access rights.
6. Integrity: validation, referential integrity, transactions, recovery.

Features of a DBMS

- **Query processor:** The Query Processor is responsible for extracting information from the database based on queries specified by SQL or other query system.

- **Schema manager** : The Schema Management facility is responsible for the maintenance of the database's self-describing information. An essential part of this is the definition of the table designs.

Features of a DBMS

- **Forms generator**: A form provides a user-friendly interface to the database, enabling the display of query results and also the management of input transactions.

- **Report generator:** Produces formatted results from the database, usually destined for printed output.

Data Dictionary

- A Data Dictionary is a centralised repository used to record all information about a database including the names of all tables, the schemas for each table, the location of tables, view definitions, details about indexes, access rights, etc.

- The data dictionary can be held within the database system itself or may be in separate database tables which are accessible by the database engine and users of the system.

Choice of Database

A number of factors can be considered in making a choice of database for a new application:

- Scale: The maximum number of users, maximum file space, etc.
- Performance: The number of transactions per hour that can be handled.
- Support for data-types: The range of data (text, numerical, graphical etc.) that can be stored and utilised.

Choice of Database

- Connectivity: Support for the accessing of other database or file systems.
- Processing complexity: The nature of the tasks that the system is intended to support.

SAMPLE DBMS

MS Access

- Directed at small businesses and advanced end-users
- 'Bundled' within the Microsoft Office suite.
- Fully-featured powerful database engine supporting multi-user working with record locking, transactions and constraints including referential integrity.
- User interface provides a graphical 'point and click' style environment for table, form, report and macro design.
- Underlying programming environment based on VBA (Visual Basic for Applications) modules.

Oracle

- Major player in the large-scale enterprise market
- Originally developed by a company called Relational Software Inc. in 1979.
- Evolved through many versions, each adding to the facilities and the system performance.
- In 1988, introduced procedural language PL/SQL in version 6
- In 1999, object-oriented features added in version 8
- In 2001, the ability to read and write XML.

MySQL

- Powerful, multi-user DBMS marketed by the Swedish company MySQL AB.
- The first formal release of the product was in 1996.
- Available as an Open Source product and versions are available free of charge
- Commonly associated with other Open Source products such as the Linux operating system, Apache web server, the web language PHP and other languages Perl and Python.
- These products can potentially provide cheaper implementations of database and web systems.

DESIGN OF TABLES

ATTRIBUTE DESIGN

Attribute Design

Choosing the data type

The data-type assigned to a table column determines four characteristics:

1. **the storage mode:** the amount of storage space used and the internal representation
2. **the behaviour of the data item on input:** the acceptable formats for entering the data and the interpretation of the data by the database
3. **the behaviour of the data item on output:** how the data is displayed on output.
4. **the permissible processing operations on the data:** e.g. arithmetic on numerical items

Attribute Design: Common Data-types

Common data-types

- **Text:** character data; letters, numerical digits, special symbols, etc. based on standard character sets such as ASCII or Unicode.
- **Numeric:** numerical values, either integer or real (floating point) numbers, with varying size and precision.
- **Counter:** System-generated serial sequence of numbers, often used to create primary key values.
- **Date/Time:** Date and time values.

Attribute Design: Common Data-types

- **Boolean:** Logical value which can be interpreted as any pair of values, e.g. true/false, 1/0, Yes/No.

- **Binary:** Set of binary data, held as an unstructured item. Often used to store multimedia data.

- **Object:** Binary data in standard object-based format such as OLE (Object Linking and Embedding) or COM ()

Attribute Design: Text Characteristics

- A maximum length must be specified based on an understanding of the data
- Systems usually provide a variable length representation which only stores the actual characters required, e.g. VARCHAR in SQL.
- Data-types are available for storing large volume text, e.g. CLOB (Character Large Object) in Oracle.

Attribute Design: Numerical Types

- Major division is between integer (whole numbers) and floating point (real numbers)
- Integers are exact but limited in range of values represented.
- Floating point values can represent a much larger range of values but with limited precision (i.e. digits of accuracy)

Attribute Design: Numerical Types

Data Type Name	Range of values	Storage
Byte	0 to 255, integer.	One byte
Integer	-32,768 to +32,767, integer.	Two bytes
Long Integer	± 2 billion, approx, integer.	Four bytes
Single	$\pm$ 3.4 x $10^{\pm 38}$, approx, floating point, six digit precision	Four bytes
Double	$\pm$ 1.8 x $10^{\pm 308}$, approx, floating point, ten digits precision.	Eight bytes

Attribute Design: Date/Time

- Handling of date and time information in databases is quite complex due to wide range of input and output formats.

- Possible date formats for 18th January 2007 are:
  - 18-01-07 and 18/01/07 (Europe),
  - 01-18-07 and 01/18/07 (US)
  - 18-Jan-07 (unambiguous)
  - 06-06-07 (ambiguous)

- Interpretation of such dates will depend on the international settings of your system.

INDEXING

Principles of Indexing

- Indexes are used to speed up accessing of database tables.
- They are an implementation requirement of practical systems rather than a theoretical feature.
- In effect, an index holds all the values of a specified column or columns of a table, together with the corresponding disk addresses of records with those values.

Principles of Indexing

- The following table is used to illustrate indexes; the 'Record No' is used simply to represent the physical position of the row.

Record No.	Customer No	Customer Name	Town
1	CD1234	Jones	Glasgow
2	AB3344	Smith	London
3	ZZ8811	Anderson	Belfast
4	RT0189	Campbell	London
5	FN2178	Harper	London
6	BC0012	Collins	Belfast

Principles of Indexing

- It is possible to build indexes for one or more fields of the file.

- This enables fast access to the data based on a known value of the chosen field.

- Fields used as indexes are often referred to as **index keys**. For example, a Customer No key would *conceptually* look as shown below:

Principles of Indexing

Customer No	Record Number
AB3344	2
CD1234	1
BC0012	6
FN2178	5
RT0189	4
ZZ8811	3

- The table is shown in dashed lines to avoid confusion with a normal table.
- The key values enable the physical position of the corresponding row to be found by the DBMS.
- In practice, a physical disk address would be used instead of a record number.

Rationale of Indexing

- The rationale for using indexes is that the index table would typically be much smaller than the data table, could be held substantially in main memory and hence can be searched more quickly.

- In the absence of an index, a search for a particular column value would necessitate a serial read of the entire data table.

- In fact, in practice, the index is structured in such a way as to further reduce the time to find a particular value.

Rationale of Indexing

- There are two separate ways in which an index helps:

1. searching rapidly for a single value
2. presenting the table in a specified order

Choosing Indexes

- There are overheads associated with maintaining indexes so decisions have to be made about which columns to index.

- The following categories of column should be indexed:

  - Primary keys
  - Foreign keys
  - Columns involved in GROUP BY or ORDER BY queries
  - Columns referred to in selection criteria of commonly-used queries.

Indexing Overheads

- The DBMS incurs certain overheads in the maintenance of indexes.

- For instance, the addition of a new row to the table will necessitate the update of every index for that table: if three columns are indexed three indexes are updated.

- Hence, the indexes must be chosen carefully to balance the speed of access with the overheads index update.

Indexing Overheads

- The indexes also occupy storage space which must be considered.

- Note that the updating of indexes is accomplished automatically by the DBMS.

- Also the query interpreter (e.g. SQL) will automatically utilise available indexes.

THANKS!

Any questions?

You can find me at elielkeelson@gmail.com &
ekeelson@knust.edu.gh