Table of Contents

# C1: Information Models

Information Models

| Information Management | Information System | Database Management System (DBMS - part of information system) |
|---|---|---|
| • capture, digitization数码化, representation, organization, transformation, and presentation of information<br>• Algorithms for efficient and effective access & update stored information (c6,7)<br>• Data modelling and abstraction (c3,4,5)<br>• Physical file storage techniques (c7)<br>• Information security, privacy, integrity and protection in a shared environment | Transform data to information then generate knowledge that can be used for decision making | • Unstructured Data<br>• OLTP & Parallel Query<br>  ○ Online transaction processing - OLTP: captures, stores, processes data from transaction in real time<br>• OLAP & Analytics<br>  ○ Online analytical processing - OLAP: uses complex queries to analyze aggregate historical data<br>• Data extract, Transform, Load<br>  ○ Eg.: A company extracts customer orders from multiple branches, transforms the format into a standard one, and loads it into a centralized system to analyze sales trends. |

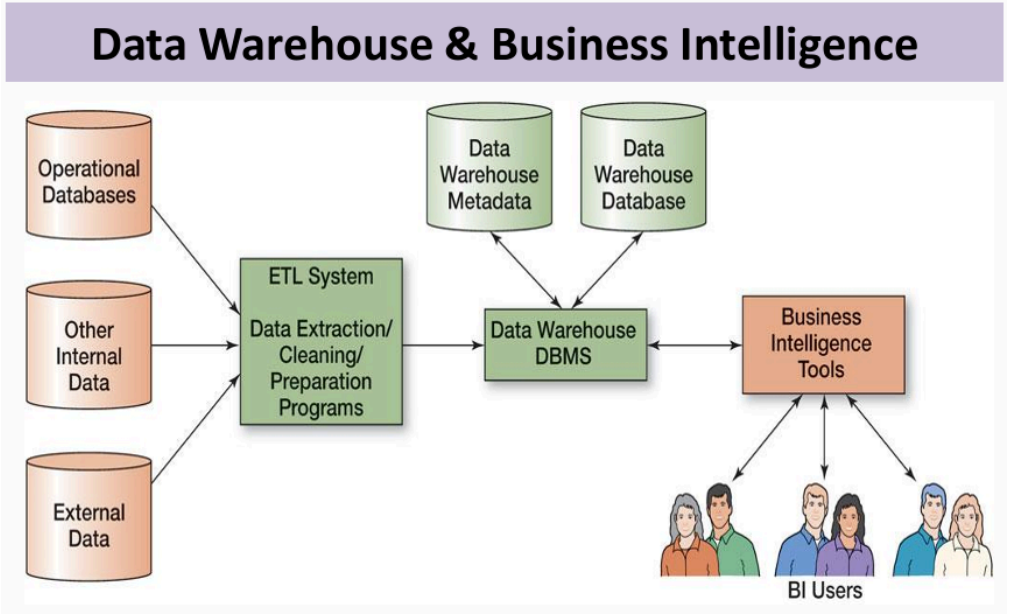
## Data, Information, Knowledge, Wisdom

| | |
|---|---|
| Data | Raw, non-summarized, unanalyzed facts and figures |
| Information | Data converted to meaningful and useful context for receiver (answers who, what, where, when) |

| | |
|---|---|
| Knowledge | Gained from information that can be used to make decision (answers how} |
| Wisdom智慧 | knowledge with insight洞察力 |

Examples

- Covid-19:
  - Data: cases number, death rate, test results (2020-03-01: 450 cases)
  - Information: a graph or table that indicates death rate per month or how daily cases are rising in different regions
  - Knowledge: Governments understand and use the data to take actions like create policies Covid-19 (travel restriction, social distance...)
  - Wisdom: Learning from Covid-19, choose to invest for better health system that can handle future outbreaks




ETL System

- Takes and converts large volumes of raw data for analysis
- Loads data in to data warehouse

Data warehouse

- Database designed to business intelligence activities (make faster and more informed business decisions)

## Categorization of Information

### 1. Source

- Primary Information: Directly obtained from the original source.
- Secondary Information: Processed from primary sources or second-hand versions.
- Internal Information: Derived from various sources within the company, such as different departments.
- External Information: Gathered outside the company, through methods like interviewing customers or examining published data.

### 2. Nature

- Qualitative Information: Non-numeric data that describes qualities and characteristics.
- Quantitative Information: Numeric data that can be measured and analyzed statistically.

### 3. Level

- Strategic Information: Used for long-term decisions and planning.
    - Examples: identify new markets, plan growth, reallocate resources
- Tactical Information: Used for tactical planning and decision-making within the guidelines set by the strategic plan.
    - Examples: Choosing suppliers, revising staffing levels, forecasting sales, inventory, and cash; preparing budgets.
- Operational Information: Used for operational planning based on tactical plans.
    - Examples: Correcting order delays, scheduling employees, finding production bottlenecks, monitoring resource usage.

#### Level of Information - Management Hierarchy

Top (Strategic)

- External data sources and summarized, tactical databases

Middle (Tactical)

- Summarized, integrated operational database

Lower (Operational)

- Individual operational databases

*Transaction Databases

- like human resource databases, inventory databases, production databases, etc.
- These databases store and manage data related to specific operational functions.

### 4. Time

- Historic Information: Gathered and stored over time, allowing for comparison between past and present activities, and identifying trends.
- Present Information: Created from activities during the current work period.
- Future Information: Created using present and historic information to predict future activities, trends, and events.

## Information Capturing:

- OCR, Barcode
- RFID
- Document Imaging
- The WWW (unstructured)
    - Twitter, Blogs, Social Networks, Web pages (Requires Big Data Analytics)

## Big Data Analytics

- Politics: Predicting election results.
- Business: Targeted social media advertising.
- Healthcare: Identifying epidemics and improving market efficiencies in delivery systems.

### Definition of Big Data

- Big data refers to data that exist in very large volumes and many different varieties (data types) and that need to be processed at a very high velocity (speed).

## Metadata/Schema Association with Data

- Metadata: Information about another set of data, literally "data about data."
- Eg.: library catalog, contains data about the data in books

- Data Dictionary: A collection of definitions and descriptions of the data elements used in a system.
    - Example fields: Name, Type, Length, Minimum, Maximum, Description, Source.

| Name | Type | Length | Min | Max | Description | Source |
|---|---|---|---|---|---|---|
| Course | Alphanumeric | 30 | | | Course I D and name | Academic Unit |
| Section | Integer | 1 | 1 | 9 | Section number | Registrar |
| Semester | Alphanumeric | 10 | | | Semester and year | Registrar |
| Name | Alphanumeric | 30 | | | Student name | Student I S |
| I D | Integer | 9 | | | Student I D (S S N) | Student I S |
| Major | Alphanumeric | 4 | | | Student major | Student I S |
| G P A | Decimal | 3 | 0.0 | 4.0 | Student grade point average | Academic Unit |

## Indexing for Fast Access

- Types of Indexing: Hashed files, B+-trees, bit-maps, etc.
- Database Efficiency: Indexing improves database efficiency and tuning.
- Queries:
    - Procedural queries are cumbersome繁琐 and prone to error容易出错.
    - Declarative queries (e.g., SQL) only require stating what you need, not how to get it.
    - Navigational queries involve knowing where to find something, such as a word, name, or brand associated with a website.

### ✅ Summary Table:

| Type | Focus On | Complexity | Example |
|---|---|---|---|
| **Procedural** | How to do it | High | `for` loop to filter data manually |
| **Declarative** | What you want | Low | `SELECT name FROM people...` |
| **Navigational** | Where it is located | Medium | Google "YouTube" → click homepage |

## Information Security (CIA)

- Confidentiality: Preserving authorized restrictions on information access and disclosure.
- Integrity: Guarding against improper information modification or destruction, ensuring information authenticity and non-repudiation.
- Availability: Ensuring timely and reliable access to and use of information.

### Threats to Information Security

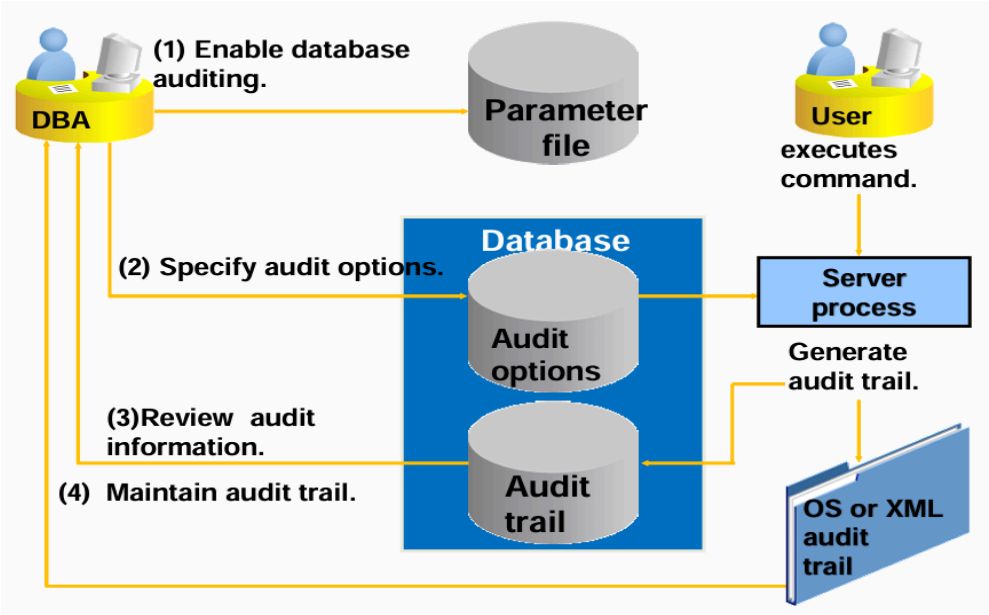
- Accidental Losses: Due to human error, software failure, hardware failure.
- Theft and Fraud.
- Loss of Privacy or Confidentiality: Loss of privacy (personal data), loss of confidentiality (corporate data).
- Loss of Data Integrity.
- Loss of Availability: Disruption of access to information or use of information (e.g., through sabotage).

### Backup & Recovery

- Physical Security: Computerized & manual policies and procedures, physical barriers, locks, security guards.

- Technical Security: Encryption, authentication, authorization, audit trace.

## Database Audit Trace




1. Enable database auditing.
2. Specify audit options.
3. Review audit information.
4. Maintain audit trail.

## Information Assurance

- Definition: Measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation.
- Nonrepudiation: The assurance that someone cannot deny something.

# C2: Database Systems

## Traditional Database Approach

### Disadvantages (5)

1. Program-Data Dependence - All programs maintain metadata for each file they use
2. Data Redundancy (duplication of data) - Different systems/programs have separate copies of the same data
3. Limited Data Sharing - No centralized control of data
4. Lengthy Development Times - Programmers must design their own file formats
5. Excessive Program Maintenance - 80% of the information systems budget

### Problems with Data Dependency

(In traditional systems (before DBMS), each program manages its own data separately. The data format, structure, and access logic are tightly connected to the program — this creates data dependency.)

- Each application program
  - data will need to maintain separately by programmer
  - needs to include code for metadata of each file
  - have its own processing routines for reading, inserting, updating and deleting data
- Lack of central control
- Non-standard file formats

Cause:

➔ waste of space as duplicate data
➔ maintain headaches
➔ inconsistencies when data changes in one file
➔ compromises data integrity 损害数据完整性

## Database Management System (DBMS)

DBMS - software system that enables users to define, create, maintain, and control access to the database.

### Function of DBMS (7)

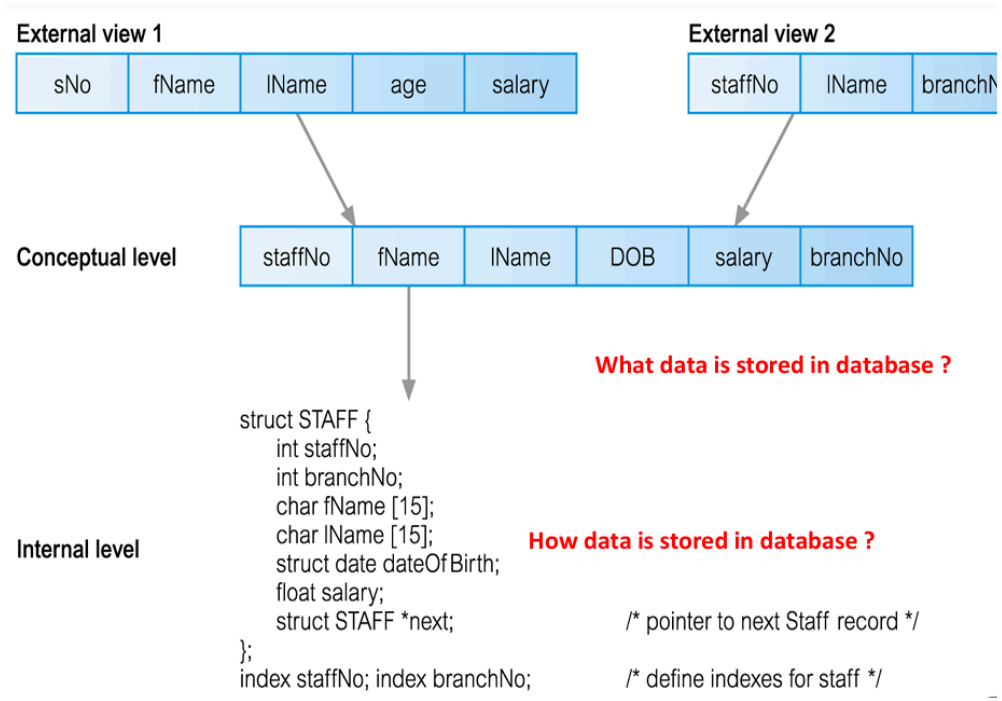
1. Store, Update and Retrieve Data
2. Concurrency Control Services 同步控制
3. Backup and Recovery
4. Security
5. Integrity Services
    a. Types Of Integrity Constraint  In Oracle
6. Transaction Support
7. User-Accessible System Catalogue (Data Dictionary Management)

### Components of DBMS

1. DML preprocessor
2. Query processor
3. Database manager
    a. Authorization control
    b. Command processor
    c. Integrity checker
    d. Query optimizer
    e. Transaction manager
    f. Scheduler
    g. Recovery manager
    h. Buffer manager
4. DDL Compiler
5. Catalog Manager

## ANSI-SPARC Three-level architecture (3)

Abstract design standard for a DBMS.




1. **External level**
    a. Users' view of database
    b. Describes part of database that is relevant to a particular user
2. **Conceptual level**
    a. Community view of the database
    b. Describe what data is stored in database and relationships among the data
3. **Internal level**
    a. Physical representation of the database on the computer
    b. Describes how the data is stored in the database

- Objectives:

Data independence (2)

1. Logical Data independence
2. Physical Data independence

## Multi-user DBMS Architectures (3)

1. Teleprocessing
2. File-Server
3. Client-server
    a. Traditional Two-Tier Client-Server
    b. Three-Tier Client-Server

c. N-Tier Client-Server Architectures
d. Transaction Processing Monitors
    i. How TP Monitor provides for consistent environment? (5)
4. Distributed DBMSs

# C3: Data Modeling

## Data Modeling

- first step in designing database, create specific data model for an information system
- Facilitate interaction among designer, programmer and the end user
- Foster improved understanding of the organization for which database design is developed
    - Give overall view of the db
    - Organize data for various users
    - Abstraction for the creation of good database
- Describing
    - Structural part
    - Manipulative part - types of operation allowed on the data
    - Set of integrity rules - ensure data is accurate

https://prnt.sc/A3Rl631XLa6W

External model
- End-user view of the data environment
- Can be many different models

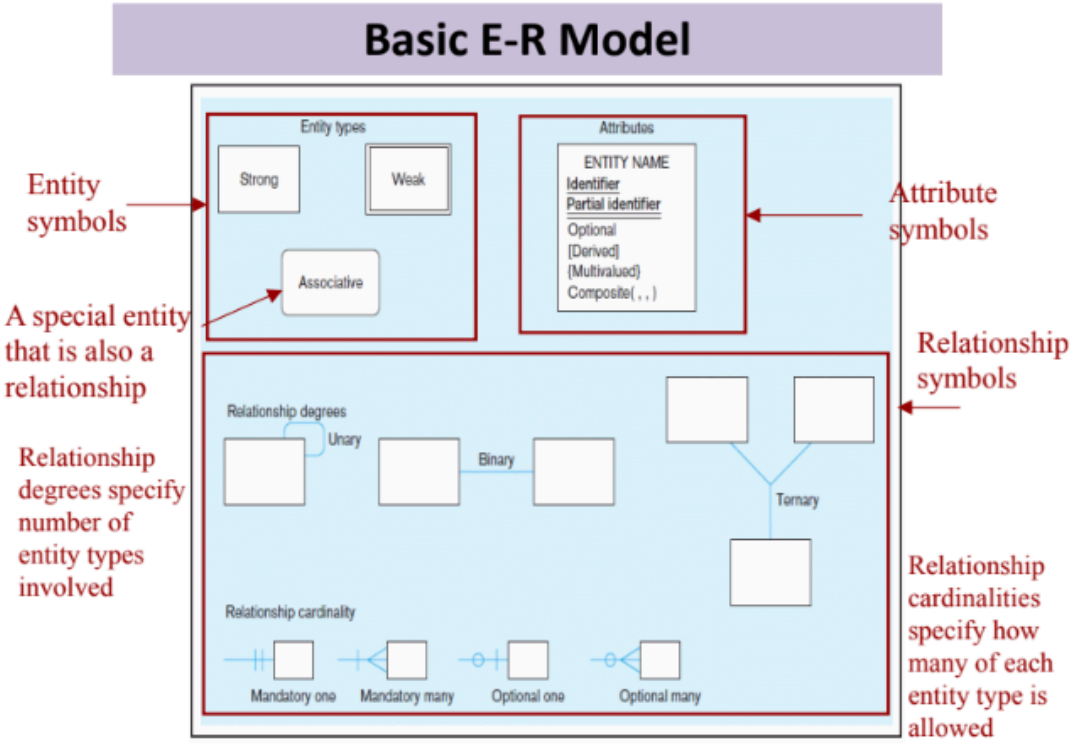
Conceptual model
- Describes what data to be stored (used in enterprise, independent of all physical considerations)
- Describes the relationships among data

Internal model
- How data stored
- Complex low-level structures described in detail

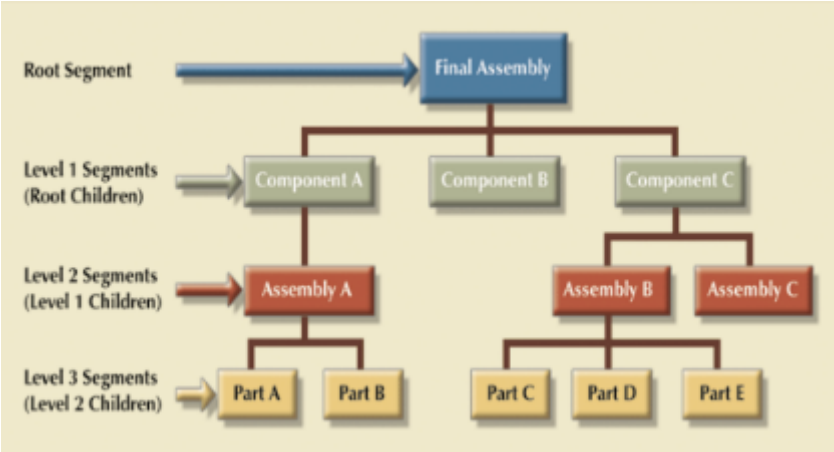
Three phases of database design:
- Conceptual database design
- Logical database design

- Physical database design.

## The Entity-Relationship Model

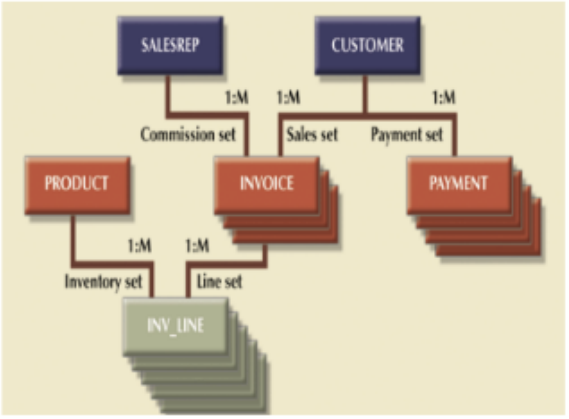



## Early Data Models




## Hierarchical model

Represented by an upside-down “tree”
It contains Levels or Segments
It depicts描绘 a set of one-to-many (1:M) relationships




**supports many-to-many relationship**

Network model
Created to represent complex data relationships more effectively than the hierarchical model, to improve database performance and to impose database standard.

# C4: Relational Model

Relation Algebra

| | | |
|---|---|---|
| Selection | Row | |
| Projection | Column | |
| Cartesian product | | **Example 4.7** Equijoin operation<br><br>*List the names and comments of all clients who have viewed a property for rent.*<br><br>In Example 4.6 we used the Cartesian product and Selection operations to obtain this list. However, the same result is obtained using the Equijoin operation:<br><br>$(\Pi_{clientNo,\ fName,\ lName}(Client)) \bowtie_{Client.clientNo = Viewing.clientNo} (\Pi_{clientNo,\ propertyNo,\ comment}(Viewing))$<br><br>**Example 4.8** Natural join operation<br><br>*List the names and comments of all clients who have viewed a property for rent.*<br><br>In Example 4.7 we used the Equijoin to produce this list, but the resulting relation had two occurrences of the join attribute clientNo. We can use the Natural join to remove one occurrence of the clientNo attribute:<br><br>$(\Pi_{clientNo,\ fName,\ lName}(Client)) \bowtie (\Pi_{clientNo,\ propertyNo,\ comment}(Viewing))$ |
| Union | OR | |
| Set difference | ~but no, ~EXCEPT | |
| Intersection | ~both + at least one, ~INTERSECT | |
| Join | Natural | |

| | | |
|---|---|---|
| | Left outer<br>Right outer<br>Semi | |
| Division | ~who xxx all | |
| Aggregate Operation | COUNT<br>SUM<br>AVG<br>MIN<br>MAX | **Example 4.12** Aggregate operations<br><br>*(a) How many properties cost more than £350 per month to rent?*<br><br>We can use the aggregate function COUNT to produce the relation R shown in Figure 4.13(a) as follows:<br><br>$\rho_R(\text{myCount})\ \Im_{\text{COUNT propertyNo}}(\sigma_{\text{rent} > 350}(\text{PropertyForRent}))$<br><br>*(b) Find the minimum, maximum, and average staff salary.*<br><br>We can use the aggregate functions, MIN, MAX, and AVERAGE, to produce the relation R shown in Figure 4.13(b) as follows:<br><br>$\rho_R(\text{myMin, myMax, myAverage})\ \Im_{\text{MIN salary, MAX salary, AVERAGE salary}}(\text{Staff})$<br><br>**Example 4.13** Grouping operation<br><br>*Find the number of staff working in each branch and the sum of their salaries.*<br><br>We first need to group tuples according to the branch number, branchNo, and then use the aggregate functions COUNT and SUM to produce the required relation. The relational algebra expression is as follows:<br><br>$\rho_R(\text{branchNo, myCount, mySum})_{\text{branchNo}}\ \Im_{\text{COUNT staffNo, SUM salary}}(\text{Staff})$<br><br>The resulting relation is shown in Figure 4.14.<br><br>branchNo \| myCount \| mySum<br>B003 \| 3 \| 54000<br>B005 \| 2 \| 39000<br>B007 \| 1 \| 9000 |

# C5: Relational Database Design

Relational Database Design

- Find a good collection of relation schemas
- Avoid repetition of information that may lead to anomalies
- Avoid loss of information (lossy decomposition 有损分解)

Problems of Data Redundancy (Update anomalies)

1. Insertion
2. Deletion
3. Modification
    a. Same info on multiple rows

QQ: not so understand what difference and how to describe

Normalization

Functional Dependency - relationship between attributes

- vital for redesign of database
    - eliminate redundancy
    - enable systematic improvement

UNF, 1NF, 2NF, 3NF, BCNF

Lower normal forms to higher

# C6: Indexing

Create indexes with SQL.
Compare performance in retrieving information when indices are used as to when they are not used.
Explain the role of an inverted index in locating a document in a collection.

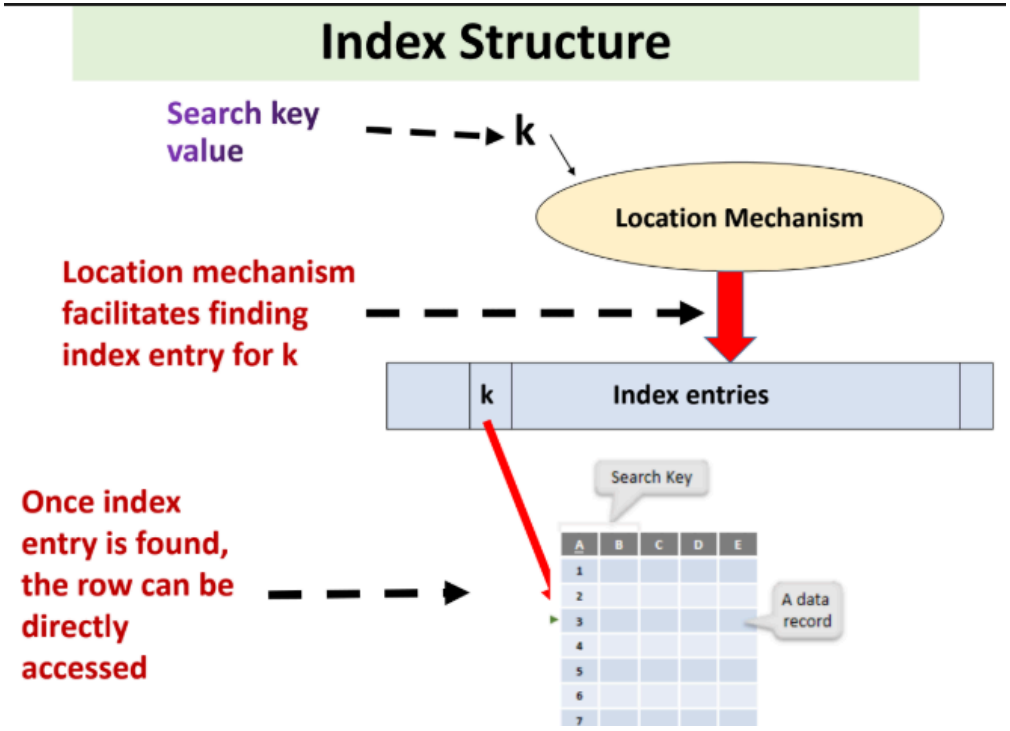
## Indexes

Special lookup查找 data structures
search engine can use to speed up data retrieval
Like book index and book catalog in library

- CREATE [UNIQUE] INDEX indexName ON TableName (columnName [ ASC | DESC ])
- DROP INDEX <index-name>

## Rules for using Indexes

1. Use on larger table
2. Index the primary key of each table
3. Index search fields (frequently in WHERE clause)
4. Fields in ORDER BY and GROUP BY commands
5. When there are >100 values but not when there are <30 values for an attribute
6. Avoid use of indexes for fields with long values; perhaps compress values first
7. If key to index is used to determine location of record, use surrogate (like sequence nbr) to allow even spread in storage area
8. DBMS may have limit on number of indexes per table and number of bytes per indexed field(s)
9. Be careful of indexing attributes with null values; many DBMSs will not recognize null values in an index search




(Search Key Value: You decide to search for books by an author's name, say "Zhang San."
Location Mechanism: The library's catalog system is the location mechanism that can quickly locate the index entry corresponding to the search key value you've provided.
Index Entries: Each index entry in the catalog contains the author's name and a reference to the physical location of the books on the shelves. For instance, the index entry might indicate that books by "Zhang San" are located on Shelf 5, Level 3.
Data Record: Once the index entry is located, you can directly access the physical location to find all books written by "Zhang San.")

## Indexes and Query Performance

Why not index every column in every table?
Not practical

- Need index-maintenance processing (especially if the table has many attributes/ rows/ requires many insert update delete)

### Common Indexing Structures

- Hash indexes

- B-tree indexes
- Bitmap indexes

## Full-text Database Systems

Storing and accessing document collections (research papers..
Indexing on text content is needed

### Indexing the Web

Most search engines use inverted index

- Inverted index
  - Composed of vocabulary and occurrences
  - For each word, the index stores the document ID and the occurrences in which that word appears.
  - A query with a conjunction of several terms is evaluated by retrieving the inverted lists of the query terms one at a time and intersecting them.
- Signature files (not used for large database sizes
- Suffix trees and arrays

# C7: Physical Database Design

Explain the different techniques for placing file records on disk.
Give examples of the application of primary/clustering and secondary/clustering indexes.
Distinguish between a non-dense index and a dense index.
Use dynamic multilevel indexes using B+-trees.
Explain the theory of hashing techniques.
Use hashing to facilitate dynamic file expansion.

## Intro

Database is stored as a collection of files
Main issues addressed

- Storage media
- File organization (Many alternatives exist, each appropriate in some situations)
- Indexes

changing (delete, insert update db) requires

- Maintaining the file structures
- Updating the indexes

File organization

- Heap
- Sequential
- Hash

Indexing structures

- Hash Index (common type of index used in database)
- B+-Tree Index (good for simple and fast lookup function)
- Bitmap index (used in data warehouse applications)
- Join index (Common for data warehouse application)

## Storage and file structures

The database is stored as a collection of files. Each file is a sequence of records.

Generally, a separate file is used to store records of a relation.
Records are either fixed size or variable size

- Variable-length records arise in database systems in several ways:
    - One or more fields have variable length
    - One or more fields are repeating
    - One or more fields are optional
    - File contains records of different types

Method of arranging a file of records on external storage.

- Heap (unordered) files: Records are placed on disk in no particular order. New records are inserted at the end of the file.
- Sequential (ordered) files: Records are ordered by the value of a specified field.
- Hash files: Records are placed on disk according to a hash function.

1. Index Evaluation Metrics

- Access time for:
    - Equality searches – records with a specified value in an attribute
    - Range searches – records with an attribute value falling within a specified range.
- Insertion time – Time to find correct place to insert the new data item + update the index structure
- Deletion time – Time to find the item to be deleted + update the index structure
- Space overhead – Space to store the index structure

---

2.1 Classification of Indexes
Clustering vs Non-Clustering

- Clustering index: ordering of index matches ordering of values of search key attribute in file

Dense vs Sparse Index

- Dense index: every value of the search key attribute found in the table also appears in the index

---

Dense Index

- Dense index: an index record appears for every search-key value in the file

---

Sparse Index

- Sparse Index: contains index records for only some search-key values.
- To locate a record with search-key value K, we:
    - Find index record with largest search-key value < K
    - Search file sequentially starting at the record to which the index record points
- Compared to dense indices:
    - Less space and less maintenance overhead for insertions and deletions.
    - With dense index, when a file is modified, every index on the file must be updated.
    - Generally slower than dense index for locating records.
    - Good tradeoff: sparse index with an index entry for every block in file, corresponding to least search-key value in the block.

---

Non-Clustering Index

- Frequently, one wants to find all the records whose values in a certain field and the file is not ordered on the field.

- Eg 1: In the instructor relation stored sequentially by ID, find all instructors in a particular department
- Eg 2: find all instructors with a specified salary or with salary in a specified range of values

---

Clustering and Non-Clustering Indices

- A non-clustering index must have an index record for every search-key value.
  - The index record points to a bucket that contains pointers to all the actual records with that particular search-key value.
- Bucket structure is used if search key is not a primary key, and file is not sorted in search key order.
- Clustering indices can be sparse - since it is always possible to find records with intermediate search-key values
- Non-clustering indices have to be dense.
- Clustering index: ordering of index matches ordering of values of search key attribute in file

Comparison

- Sequential scan using clustering index is efficient
- Sequential scan using a non-clustering index is expensive – each record access may fetch a new block from disk

---

3. Multilevel Index

- Imagine this:
  - A file 100,000 records, 10 records in one block

- Sparse index => 1 index record per block, how many records will the index have?
  - 100000/10 blocks to store the data records.
  - There will be 10000 index records.
  - Index will require 10000/100 = 100 blocks

Multilevel Index

- If an index does not fit in memory, access becomes expensive.
- To reduce number of disk accesses to index records, construct:
  - outer index – a sparse index on main index
  - inner index – the main index file
- If even outer index is too large to fit in main memory, yet another level of index can be created, and so on.
- Indices at all levels must be updated on insertion or deletion from the file.

---

4. B+-Tree Index Files

- A B+-tree must satisfy the following properties:
  - All paths from root to leaf are of the same length
  - Each node that is not a root or a leaf (nonleaf node) has between ⌈n/2⌉ and n children where n is the maximum number of pointers per node. n is fixed for a particular tree

- A leaf node has between ⌈(n – 1)/2⌉ and n – 1 values
- Special cases: if the root is not a leaf, it has at least 2 children. If the root is a leaf (that is, there are no other nodes in the tree), it can have between 0 and n-1 values.

---

Example of B+-Tree of Order 6 (n=6)

- Leaf nodes must have between 3 and 5 values (⌈(n–1)/2⌉ and n–1, with n = 6)
- Non-leaf nodes other than root must have between 3 and 6 children (⌈(n/2⌉ and n with n =6)
- Root must have at least 2 children

Balanced Tree

- A B+-tree needs to be BALANCED, meaning that all of its leaf nodes are at the same level.

Unbalanced Tree

- Keeping a tree balanced is important to guarantee that no node will be very high levels and hence require many block accesses during a tree search

---

B+-Tree Node Structure

- Ki are the search-key values
- Pi are pointers to children (for non-leaf nodes) or pointers to records or buckets of records (for leaf nodes)
- The search-keys in a node are ordered: K1 < K2 < K3 < ... < Kn–1

---

Leaf Nodes in B+-Trees

- For i = 1, 2, ..., n–1, pointer Pi points to a file record with search-key value Ki
- If Li, Lj are leaf nodes and i < j, Li's search-key values are less than or equal to Lj's search-key values
- Pn points to next leaf node in search-key order

---

Non-Leaf Nodes in B+-Trees

- For a non-leaf node with n pointers:
    - All the search-keys in the subtree to which Pi points are less than Ki
    - All the search-keys in the subtree to which Pi points are greater than or equal to Ki1

---

Queries on B+-Trees

- Find all records with a search-key value of k.
    - Start with the root node
    - Examine the node for the smallest search-key value > k
        - If such a value exists, assume it is Ki. Then follow Pi
        - Else if k ≥ Km-1, follow Pm
    - Repeat the procedure on the next node
    - Eventually reach a leaf node. If key Ki = k, follow pointer Pi to the record/bucket. Else, no record with k

---

Updates on B+-Trees: Insertion

- Find the leaf node for the search-key value
    - If value exists, add to file/bucket
    - Else, insert (key, pointer) into leaf node
        - If node full → split
            - Sort n pairs
            - Place first ⌈n/2⌉ in original node, rest in new
            - Insert (least key in new node, pointer) to parent
        - Split may propagate up, possibly increasing tree height by 1

---

Updates on B+-Trees: Deletion

- Find and remove record from file and bucket (if any)
- Remove (key, pointer) from leaf node
- If underfull:
    - If sibling + current fit in one → merge into left
        - Delete pointer from parent recursively

    - Else → redistribute with sibling and update parent key

- May propagate upwards
    - If root has one pointer, delete root and make child new root

---

5. Hash Index

- Hashing can be used for index-structure creation
- A hash index organizes search keys with pointers into a hash file structure

Static Hashing

- Function h maps search-key values to fixed bucket addresses

Bucket Overflow

- Causes:
    - Too few buckets
    - Skew in distribution
- Handling:
    - Overflow buckets chained as linked list

Deficiencies of Static Hashing

- Growth → performance degrades (too many overflows)

- Preallocated space wastes storage initially
- Shrink → space wasted

---

Dynamic Hashing

- Allows hash function to be modified dynamically
- Extendable hashing...

# C8: Transaction Processing

1. Database Software Security Features
   - Views
   - Integrity controls
   - Authorization
   - Encryption
   - Authentication schemes
   - Backup & recovery

---

Views
- Views
  – Subset of the database that is presented to one or more users
  – User can be given access privilege to view without allowing access privilege to underlying tables

sql
**复制编辑**
```
CREATE VIEW low_stock_view AS
SELECT *
FROM products
WHERE quantityInStock < 50
WITH READ ONLY CONSTRAINT low_stock_readOnly;
```

---

Integrity Controls

- Ensures that data entered into the database is accurate, valid, and consistent.
- Three basic types of integrity controls are:
    - Entity integrity: Not allowing multiple rows to have the same identity within a table.
        - Primary key

Domain integrity: Restricting data to predefined types.

sql
**复制编辑**

```
CREATE DOMAIN PriceChange AS DECIMAL
CHECK (VALUE BETWEEN .001 and .15);
```

    - 
    - Referential integrity: Requiring the existence of a related row in another table.

---

Referential Integrity

- If FK contains a value, that value must match a CK in parent table.
- Actions using ON UPDATE and ON DELETE:
    - CASCADE
    - SET DEFAULT
    - SET NULL

- NO ACTION

sql
复制编辑

```
CREATE TABLE Staff (
  StaffNo VARCHAR(4) NOT NULL,
  ...
  BranchNo VARCHAR(4) DEFAULT 'B005',
  PRIMARY KEY (StaffNo),
  FOREIGN KEY (BranchNo) REFERENCES Branch
    ON UPDATE CASCADE
    ON DELETE SET DEFAULT
);
```

---

Referential Integrity Actions Explained

- CASCADE: Delete parent row and matching child rows.
- SET NULL: Set child FK columns to NULL (if allowed).
- SET DEFAULT: Set FK in child to default value.
- NO ACTION: Reject deletion from parent.

---

Authorization

- DBMS controls to restrict:
  - Access to data

    - Actions users can take

- Oracle privileges:
    - Can be granted at database/table level
    - INSERT, UPDATE can be granted at column level
- Authorization Matrix:
    - Subjects
    - Objects
    - Actions
    - Constraints

---

Authorization (continued)

- Relation owner is granted all privileges.
- Privileges can be passed using WITH GRANT OPTION.
- To create a view, SELECT privilege is needed on all involved relations.

sql
复制编辑
```
CREATE VIEW BEMPLOYEE AS
SELECT NAME, BDATE, ADDRESS
```

```
FROM EMPLOYEE
WHERE DNO = 5;

GRANT SELECT ON BEMPLOYEE TO B
WITH GRANT OPTION;
```

---

Encryption

- Scrambling of data for unreadability.
- SSL is a popular encryption scheme for TCP/IP.

---

Authentication Schemes

- Goal: Positive identification of user
- Methods:
    - Passwords
    - Two-factor (e.g. smart card + PIN)
    - Three-factor (e.g. biometric + card + PIN)
    - Biometric devices
    - Third-party mediated (e.g. digital certificates)

---

2. Transaction Support

- Transaction: Logical unit of work that reads or updates the database.

Boundaries of transaction:

- BEGIN TRANSACTION
- COMMIT: Successful completion
- ROLLBACK: Abort and restore to previous state

Example (ATM):
sql
复制编辑
BEGIN TRANSACTION
...
If balance is sufficient then
  UPDATE account
  INSERT history
...
On Error: ROLLBACK
COMMIT

---

ACID Properties

- Atomicity: All or nothing
- Consistency: Valid state transitions
- Isolation: Incomplete changes not visible

- Durability: Committed changes are permanent

---

2.1 Concurrency Control

- Manages simultaneous DB operations
- Prevents interference from multiple users

---

Concurrency Problems

- Lost Update Problem
- Uncommitted Dependency (Dirty Read)
- Inconsistent Analysis Problem

---

Examples

- Lost Update:
    - T2's update lost because T1 reads before T2 completes.
- Dirty Read:
    - T3 reads uncommitted changes from T4.
- Inconsistent Analysis:
    - T6 reads while T5 is updating.

---

2.2 Serializability

- Schedule: Sequence of operations
- Serial: One transaction at a time
- Non-serial: Interleaved operations
- Serializable: Non-serial but equivalent to a serial schedule

---

Rules of Conflict

- Only conflicts if one writes and another reads/writes the same item.

---

2.3 Concurrency Control Techniques

2.3.1 Locking

- Shared (read) lock: Allows reading only
- Exclusive (write) lock: Allows read/write

Lock Compatibility Matrix:

| | Shared | Exclusive |
|---|---|---|
| Shared | Yes | Wait |
| Exclusive | Wait | Wait |

---

Two-Phase Locking (2PL)

- Growing phase: Acquires locks
- Shrinking phase: Releases locks
- Prevents:
    - Lost Update
    - Dirty Read
    - Inconsistent Analysis

---

Deadlock

- Circular waiting for locks

---

Handling Deadlocks

- Timeouts: Abort after wait
- Prevention: Use timestamps:
    - Wait-Die: Older waits, younger dies
    - Wound-Wait: Older wounds younger
- Detection: Use Wait-For Graph (WFG)

---

WFG Deadlock Detection

- Node for each transaction
- Edge Ti → Tj if Ti waits on Tj
- Deadlock = cycle in WFG

---

2.3.2 Versioning

- Optimistic concurrency control
- Multiple versions of a data item
- Assumes simultaneous updates are rare

# C9: Distributed Database

- Understand characteristics of distributed database environments.
- Explore design strategies including data replication and partitioning.
- Analyze query processing and concurrency in DDBMSs.

---

2. Introduction to Distributed Database
   - Definition: A logically interrelated collection of shared data, physically distributed across a network.

Centralized vs Distributed
   - Centralized DB: Single location.
   - Distributed DB: Data stored across multiple sites.

---

3. Advantages of Distributed DBMS
   - Reliability/Availability: System remains functional even with component failures.
   - Local Control: Sites manage their own data/security.
   - Scalability: Easy to add nodes.
   - Reduced Communication Costs: Data near users.
   - Faster Queries: For locally stored data.

Disadvantages

- High Software Complexity and Cost
- Processing Overhead
- Data Integrity Challenges
- Possible Slower Query Response (if poorly distributed)

4. Types of DDBMS
- Homogeneous: Same DBMS across sites.
- Heterogeneous: Different DBMSs and data models; requires translation (e.g., via gateways).

5. Distributed Data Storage
Fragmentation/Partitioning
- Horizontal: Rows distributed across sites.
- Vertical: Columns distributed.
- Hybrid: Mix of horizontal and vertical.

Pros:
- Local optimization
- Increased efficiency and security

Cons:
- Inconsistent access speeds

- No built-in replication

---

Data Replication

- Push Replication: Source sends updates.
    - Snapshot: Periodic.
    - Near Real-Time: Immediate, often via triggers.
- Pull Replication: Receiver requests updates.

Issues:

- Data freshness
- Network load
- System heterogeneity

Pros:

- High availability
- Faster local queries
- Reduced network traffic

Cons:

- Storage and update overhead

- Consistency risks

---

Data Allocation Strategies

- Centralized: Single location.
- Partitioned: Fragments stored at different sites.
- Complete Replication: Full DB copy at all sites.
- Selective Replication: Hybrid of above.

Goals:

- Locality
- Reliability
- Load balancing
- Minimized communication

---

6. Distributed Query Processing

Example: Join Employee (site 1) and Department (site 2), return result to site 3.

- Strategy 1: Move both tables to site 3 (1,003,500 bytes).
- Strategy 2: Move Employee to site 2, then result to site 3 (1,400,000 bytes).
- Best: Move Department to site 1, join, send result (403,500 bytes).

Semijoin Strategy:

1. Transfer only join attributes.
2. Join locally.
3. Transfer reduced dataset.

Goal: Minimize network data transfer.

---

7. Transparencies in DDBMS

Types

- Distribution Transparency: Users unaware of data distribution.
    - Fragmentation: No knowledge of how data is split.
    - Location: No need to know where data is stored.
    - Replication: No awareness of multiple copies.
- Transaction Transparency: Ensures integrity across sites.
    - Concurrency & Failure Transparency
- Performance Transparency: As efficient as centralized.
- DBMS Transparency: Hides heterogeneity.

---

8. Distributed Transactions

Definition:

A transaction that accesses data from multiple sites.
Components:

- Sub-transactions: One per involved site.
- Transaction Coordinator: Oversees execution, communication, and termination.
- Local Transaction Manager: Handles logging, concurrency, and recovery.

# PYQ

**Question 1**

a) Differentiate between *procedural* and *declarative* queries and provide an example for each type of queries. (5 marks)

b) Explain the **FIVE (5)** disadvantages of *traditional approach* (computer file-based system) of programming with data files. (10 marks)

c) With reference to the TAR UMT environment, illustrate and explain (using ERD, Crow's Foot notation) how the relationships between *Student*, *Programme* and *Course* entities are modelled (exclude all attributes). Many-to-many relationship need to be resolved and you must indicate the strong or weak relationship lines. (6 + 4 marks)

[Total: 25 marks]

OCT 2024 - Q1

a) Differentiate between **procedural and declarative queries** and provide an **example** for each type of queries.(5 marks)

b) Explain the **FIVE (5) disadvantages of traditional approach** (computer file-based system) of programming with data files. (10 marks)

c) With reference to the TAR UMT environment, illustrate and explain (using ERD, Crow's Foot notation) how the relationships between Student, Programme and Course entities are modelled (exclude all attributes). Many-to-many relationship need to be resolved and you must indicate the strong or weak relationship lines. (6 + 4 marks)

JAN 2025 - Q1

a) For each TAR UMT item in the following table, provide an example to differentiate between *quantitative* and *qualitative* information.

| | Item | Quantitative Information | Qualitative Information |
|---|---|---|---|
| (i) | Bus | | |
| (ii) | Building | | |
| (iii) | Tree | | |
| (iv) | Table | | |
| (v) | Projector | | |

(10 marks)

b) With reference to the TAR UMT environment, illustrate and explain (using ERD, Crow's Foot notation) how the relationships between *Lecturer*, *Student* and *Course* entities are modelled (exclude all attributes). Many-to-many relationship needs to be resolved and you must indicate the strong or weak relationship lines. (15 marks)

[Total: 25 marks]

a) For each TAR UMT item in the following table, provide an example to differentiate between quantitative and qualitative information. (10 marks)

| Item | Quantitative Information | Qualitative Information |
|---|---|---|
| (i) Bus | | |
| (ii) Building | | |
| (iii) Tree | | |
| (iv) Table | | |
| (v) Projector | | |

b) With reference to the TAR UMT environment, illustrate and explain (using ERD, Crow's Foot notation) how the relationships between Lecturer, Student and Course entities are modelled (exclude all attributes). Many-to-many relationship needs to be resolved and you must indicate the strong or weak relationship lines. (15 marks)

# (C4) Relational algebra

先判断projection还是selection (第一个拿的xxx)

有需要拿的相关的拿就join -> ⋈ (xxx.xyID = yyy.xyID (yyy) )

OCT 2024 - Q2a

**Question 2**

The *Fun Gathering Toy Shop's* database is shown as follows:

**Toy (ToyID, ToyName, ToyCategoryID*, ToyPrice, QuantityInStock, SuitableAgeGroup)**
**ToyCategory (ToyCategoryID, ToyCategoryName)**
**Customer (CustID, CustName, CustAddress, CustContact, CustGender)**
**Staff (StaffIC, StaffName, StaffAddress, StaffContact, StaffGender, StaffPosition, StaffSalary)**
**Order (OrderID, OrderDate, CustID*, StaffIC*)**
**OrderDetails (OrderID*, ToyID*, SellingPrice, Qty, Subtotal)**

Note: customer or staff gender is either 'M' for male or 'F' for female.

a) Write a ***relational algebra*** statement for each of the following questions:

(i) List out all female customers (ID, name, address and contact) who are staying at '*Setapak Centre*'. (3 marks)

(ii) List out all cashiers (IC, name and gender) whose salary is more than or equal to RM2500. (3 marks)

(iii) List out all customers (ID, name and contact) who had made an order in February 2024 and served by staff named Edwina. (6 marks)

(iv) List out the total number of toys for each toy category (ID and name). (4 marks)

Q2a)

i) $\Pi_{CustomerID, CustName, CustAddress, CustContact}(\sigma_{CustAddress = \text{"Setapak Central"} \wedge CustGender = 'F'}(Customer))$

ii) $\Pi_{StaffIC, StaffName, StaffGender}(\sigma_{StaffPosition = \text{"cashier"} \wedge StaffSalary \geq 2500}(Staff))$

iii) $\Pi_{CustID, CustName, CustGender}(Customer) \bowtie (Customer.CustID = Order.CustID(\sigma_{OrderDate \geq '1-2-2024' \wedge OrderDate \leq '29-2-2024'}(Order))) \bowtie (Order.StaffIC = Staff.StaffIC(\sigma_{StaffName = \text{"Edwina"}}(Staff)))$

iv) $\Pi_{ToyCategoryID, ToyCategoryName}(ToyCategory) \bowtie (ToyCategory.ToyCategoryID = Toy.ToyCategoryID(\Im_{COUNT\ QuantityInStock}(Toy)))$

JAN 2025 - Q2a

The *Animal Adoption System's* database is shown as follows:

**Zoo (ZooID, ZooName, SizeInAcres, OperationDate, CountryID*)**
**Country (CountryID, CountryName)**
**Animal (AnimalID, AnimalName, AnimalGender, Quantity, CategoryID*)**
**Category (CategoryID, CategoryName)**
**AnimalAdoption (ZooID*, AnimalID*, AdoptionDate, CostPerUnit, AdoptionQuantity)**

Note: animal gender is either 'M' for male, 'F' for female or 'H' for Hermaphrodite. Different animal ID will be assigned to animals based on its gender.

a) Write a ***relational algebra*** statement for each of the following questions:

(i) List out all animals (ID, name, gender and quantity) which are under the 'Cat' category. (3 marks)

(ii) List out all zoos (ID, name, size and operation date) within Malaysia. (3 marks)

(iii) List out all hermaphroditic animals (ID, name and quantity) which are adopted by 'Safari Zoo' in the month of May 2024. (6 marks)

(iv) List out the total quantity of animals for each animal category (ID and name). (4 marks)

---

Q2a)

i) $\Pi_{AnimalID, AnimalName, AnimalGender, Quantity}(Animal)$
$\bowtie (Animal.CategoryID = Category.CategoryID$
$(\sigma_{categoryname = "Cat"}(Category)))$

ii) $\Pi_{ZooID, ZooName, SizeInAcres, OperationDate}(Zoo)$
$\bowtie [Zoo.CountryID = Country.CountryID$
$(\sigma_{countryName = "Malaysia"}))$

iii) $\Pi_{AnimalID, AnimalName, Quantity}(\sigma_{AnimalGender = "H"}(Animal))$
$\bowtie (Animal.AnimalID = AnimalAdoption.AnimalID(\sigma_{AdoptionDate \geq '2024-05-01'}$
$\wedge AdoptionDate \leq '2024-05-31'(AnimalAdoption)))$
$\bowtie (AnimalAdoption.ZooID = Zoo.ZooID$
$(\sigma_{ZooName = "Safari Zoo"}(Zoo)))$

iv) $\Pi_{categoryID, categoryName}(Category)$
$\bowtie (Category.CategoryID = Animal.CategoryID$
$\Im_{COUNT\ Quantity}(Animal)$

# SQL commands based on scenarios

OCT 2024 - Q2b

b) Write the ***SQL commands*** to fulfil the requirements specified for the following scenarios:

| | Object (Table) | Access (Yes / No) | Authorisation | User | Allow other users to access (Yes / No) | |
|---|---|---|---|---|---|---|
| (i) | Toy | Yes | Read on ToyID, ToyName and ToyPrice. | All Users | No | (2 marks) |
| (ii) | Toy | Yes | Update on ToyName and ToyPrice. | Janice | No | (2 marks) |
| (iii) | Toy | Yes | Can do whatever. | Francis | Yes | (3 marks) |
| (iv) | Toy | No | No authorisation. | Janice | No | (2 marks) |

[Total: 25 marks]

(i) Access: Yes

**Authorisation**: Read on `ToyID`, `ToyName`, and `ToyPrice`
**User**: All users
**Allow other users to access**: No

```
GRANT SELECT (ToyID, ToyName, ToyPrice) ON Toy TO PUBLIC;
```

- `SELECT` allows read-only access.
- `TO PUBLIC` means all users.
- No `WITH GRANT OPTION`, so users cannot grant this permission to others.

(ii) Access: Yes

**Authorisation**: Update on `ToyName` and `ToyPrice`
**User**: Janice
**Allow other users to access**: No

```
GRANT UPDATE (ToyName, ToyPrice) ON Toy TO Janice;
```

- Allows `Janice` to update only `ToyName` and `ToyPrice`.
- No `WITH GRANT OPTION`, so she cannot give access to others.

(iii) Access: Yes

**Authorisation**: Can do whatever
**User**: Francis
**Allow other users to access**: Yes

```
GRANT ALL PRIVILEGES ON Toy TO Francis WITH GRANT OPTION;
```

- `ALL PRIVILEGES` means full control: SELECT, INSERT, UPDATE, DELETE, etc.
- `WITH GRANT OPTION` allows Francis to give permissions to other users.

(iv) Access: No

**Authorisation**: No authorisation
**User**: Janice

```
REVOKE ALL PRIVILEGES ON Toy FROM Janice;
```

- Removes any access Janice might have on the Toy table.

## 1, 2, 3NF + dependencies + 3 anomalies

OCT 2024 - Q3a

Given the **AnimalAdoption** table as follows:

| ZooID | ZooName | CountryID | CountryName | AnimalID | AnimalName | CategoryID | CategoryName | AdoptionDate | CostPerUnit | Qty |
|---|---|---|---|---|---|---|---|---|---|---|
| Z1001 | Taiping Zoo | C101 | Malaysia | A1433 | White Tiger | C008 | Cat | 08/01/2023 | 15000.00 | 2 |
| Z1001 | Taiping Zoo | C101 | Malaysia | A2722 | White Owl | C003 | Bird | 08/01/2023 | 3000.00 | 3 |
| Z1001 | Taiping Zoo | C101 | Malaysia | A1433 | White Tiger | C008 | Cat | 10/09/2024 | 16500.00 | 1 |
| Z1008 | Chiangmai Zoo | C102 | Thailand | A1433 | White Tiger | C008 | Cat | 08/03/2023 | 14500.00 | 2 |
| Z1008 | Chiangmai Zoo | C102 | Thailand | A2755 | Robin | C003 | Bird | 20/06/2023 | 5000.00 | 3 |
| Z1100 | Surabaya Zoo | C103 | Indonesia | A2728 | Parrot | C003 | Bird | 12/08/2023 | 7500.00 | 2 |
| Z1322 | Louis Mini Zoo | C104 | Brunei | A2755 | Robin | C003 | Bird | 11/11/2023 | 5200.00 | 2 |
| Z1322 | Louis Mini Zoo | C104 | Brunei | A2722 | White Owl | C003 | Bird | 23/01/2024 | 3200.00 | 2 |

**Table 1: Details of AnimalAdoption Table**

a) Normalise Table 1 to a set of Third Normal Form (3NF) relations. Your answer should show all the three stages of normalisation (1NF, 2NF and 3NF) by using the Database Design Language format (underline all primary keys, composite keys and use an * to indicate the foreign keys). State the functional dependency/dependencies that is/are removed from second and third Normal Form. Besides that, 1NF must be divided into repeating and non-repeating group relations from its original 1NF table. (16 marks)

b) Based on the sample data shown in the **AnimalAdoption** table above, provide a specific example for insertion, modification and deletion anomalies. (9 marks)

[Total: 25 marks]

## B+- tree final structure with degree

OCT 2024 - Q4a

a) Based on the following set of DogID for the **Dog** table as shown in *Table 2*:

| DogID | DogName | DogGender | CostPerUnit |
|---|---|---|---|
| 501 | Pomeranian | M | 3800.00 |
| 502 | Welsh Corgi | F | 5800.00 |
| 503 | Maltese | M | 5400.00 |
| 504 | American Staffordshire Terrier | M | 6500.00 |
| 505 | Papillion | F | 5200.00 |
| 506 | Cane Corso | M | 4500.00 |

**Table 2: Dog Table**

(i) Construct a *B+-tree* final structure with degree of 3 (6 marks)

(ii) Construct a *B+-tree* final structure with degree of 4 (4 marks)

(iii) Construct a *B+-tree* final structure with degree of 5 (3 marks)

## Hash structure

OCT 2024 - Q4b

b) Suppose the hash function is h(x) = x mod 8 and each bucket can hold at most two records. Show the extendable hash structure after inserting 1, 12, 13, 7, 16, 2 and 4.

| x | 1 | 12 | 13 | 7 | 16 | 2 | 4 |
|---|---|---|---|---|---|---|---|
| h(x) | 001 | 100 | 101 | 111 | 000 | 010 | 100 |

(12 marks)