NOTES ON – DATABASE SYSTEMS **INF3703/2012**

Chapter 1 and 2 – just read as background not examined

Chapter 3: The relational Database Model

Properties:

1. A table is a two-dimensional structure composed of rows and columns
2. Each row (tuple) represents a single entity – duplicate rows are not allowed
3. Each column represent a attribute and has a name
4. Each cell (column/row) contain only an atomic value – a single data value
5. All values in a column must conform to the same data format
6. Each column has a specific range of values – attribute domain
7. The order of the rows and columns in immaterial to the DBMS
8. Each table must have a attribute of combination of attributes to uniquely identify each row

Degree and cardinality

- Cardinality of a table is the N rows (4 rows means cardinality of 4)
- Degree is the number of columns in the table

Keys

- A key consist of one of more attributes that can be used to identify other attributes. Examples of keys are primary, super keys, candidate keys and secondary keys.
- The role of the key is based on the concept known as **'determination'** – use a key in table A to lookup of (determine) a value in table B.
- The term **functional dependence** can be defined as the attribute A is functional dependant of B.
- We might need more than one attribute to define a unique key also called multi-attribute or **composite** key. E.g. STU_NUM, STU_NAME
- Superkey – any key that uniquely identify each row e.g. STU_NUM, STU_NAME
- Candidate key – a superkey without redundancies e.g. STU_NUM
- A foreign key (FK) is an attribute whose values match the primary key values in the related table
- Secondary key is defined as a key that is use strictly for data retrieval

Integrity rules

1. Entity integrity
    a. All primary keys are unique and no part of the primary key may be null
    b. Each row will have a unique identity and foreign key values can refer to as primary key values. E.g. No invoice has a duplicate number, or can be null, and uniquely identified by invoice number.
2. Referential integrity
    a. A foreign key may have a null entry.
    b. It is possible for an attribute not to have a corresponding value but impossible to have an invalid entry. The referential rule makes it possible to delete a row in one table whose primary key has mandatory values in another table. E.g. a customer might not have a sales representative yet but its impossible to have sales

Multiplicity – in UML model provides same info as the connectivity, cardinality and relationship in the ER model.

Indexes
An index is an orderly arrangement use to logically access rows in a table using an index key. Use to retrieve data more efficiently. Also to retrieve data orderly e.g. creating a index on CUS_LAST_NAME will retrieve data alphabetically. Add an index to more than one attribute such as VENDOR_NAME and CODE, will retrieve the code in sequence within the vendor alphabetically.

CODD's RELATIONAL DB RULES – page 101

Chapter 5 – Design Concepts

The ER Model consist of Entities, attributes and relationships

Entities – is an object of interest to the user. In ER modelling an entity actually represents an entity set not a single occurrence. In the ERM an ENTITY corresponds to a TABLE and not a row as in the relational environment.

In the UML notation an entity is represented by a box that is subdivided into three parts:
- The name at the top in capital letters
- The middle to name and describe the attributes
- The bottom to list the methods – only used in OO Db models therefore left blank.

Generally we use the UML class diagram ERDs

Attributes – are characteristics for entities. E.g. the STUDENT entity the attributes could be STU_NAME, STU_SURNAME

Domains – is a attribute set of possible values. E.g. a domain for a numeric attribute GRADE_POINT could have a value (0,4) or SEX could have M or F

Identifiers (Primary keys) – an ERM uses identifiers to uniquely identify each entity instance. Identifiers are underlined in the ERD. Key attributes are also underlined when writing the relational schema.

TABLE NAME(KEY_ATTRIBUTE, ATTRIBUTE_2, ATTRIBUTE_3)

Composite primary keys – Ideally a primary key is composed of a single attribute, however it is possible to use a composite key. E.g. The CLASS entity could only be identified using CRS_CODE and CLASS_SECTION. The combination of both is also a proper candidate key.

Attributes could also be Simple or Composed. E.g. the composed attribute ADDRESS could be further broken down in Simple attribute STREET, CITY etc.

Multivalued attributes – a attribute could have more than one value such as CAR_COULOR. The resolve the problem we could create new attributes. CAR_TRIM_COL, CAR_BODY_COL. You need to keep the original table and create another table and adding a primary key to link the tables like CAR_REG

Multiplicity – refers to the number of instances of one entity that are associated with one instance of a related entity. E.g. LECTURE and CLASS table, a lecture can teach up to 4 classes but a class can only be taught by one lecturer. The notation will be 1..4

Existence dependence - an entity is said to be existence-dependent of it can exist in the database only when associated with another related entity. E.g. employee in the XYZ Corporation table, the employee table could not exist without the company XYZ.

Existence independent – Vendor and parts – parts could be sold by other vendors and vendors does not have to supply all the parts

Weak (non-identifying) **relationships** – if the PK of the related entity does not contain a PK component in the parent entity.

Weak **entity –** needs to meet the following conditions.
- It is existence-dependent, that is, it cannot exist without the entity with which it has a relationship
- It has a primary key that is partially or totally derived from the parent entity in the relationship

Relationship degree – indicates the number of entities or participants associated with the relationship. (see page 176)
Unary – one table
Binary – two tables
Ternary – three tables link to the main table

Developing an ER Diagram

1. Create a detailed narrative of the organisations description of operations
2. Identify the business rules based on the descriptions\
3. Identify main entities from the business rules
4. Identify all main relationships between entities and business rules
5. Develop and initial ERD
6. Determine multiplicities and the participation of all relationships. (optional or mandatory)
7. Identify the primary and foreign keys
8. Identify all attributes
9. Revise and review the ERD

Database design challenges:

- The database must conform to design standards that guide the logical structure and minimize redundancies. Design standards allow you to work with well-defined components and to evaluate interaction between them.
- High processing speed is often top priority, which may be achieved by minimizing the number of complex logical relationships.
- Timely information could be achieved through expanding the number of entities.

Chapter 6: Advanced data modelling

The extended entity relationship model (EEER) – also called enhanced entity relationship model is as a result of adding more semantic constructs to the original EER via entity supertypes, subtypes and clusters.

Entity supertype – is a generic entity type that relate to one or more subtypes. The supertype contains the common characteristics and the subtype the unique characteristics. The supertypes and subtypes are arranged by the specialization hierarchy.

Inheritance allows an entity subtype to inherit the attributes and relationships of the supertype. Subtypes can be disjoint or overlapping. The subtype discriminator is use to determine which entity subtype the supertype occurrence is related. (Figure 6.2 – page 214)

Selecting primary keys:

The following characteristics make a PK good

1. Unique values – The PK must uniquely identify each entity instance. Must be able to guarantee unique values and cannot contain nulls.
2. Nonintelligent – Should not have embedded sematic meaning. E.g. student ID number of 1234 is preferred over Smith, J,L.
3. No change over time – thus not subject to updates. E.g it a name is taken as a PK, what happens if the person gets married.
4. Preferably single-attribute – not a rule, but makes implementation of foreign keys easier.
5. Preferably numeric – easier to manage in database.
6. Security complaint – must not consist of attributes that might be considered a security risk or violation.

Surrogate primary keys are useful when there are (1) no natural key that makes a suitable PK, (2) the PK is a composite PK with multiple different data types and (3) the PK is to long.

Selecting a foreign key:

IF: one side is mandatory and one side is optional – place the PK of the entity on the mandatory side in the entity on the optional side as a FK and make the FK mandatory

IF: both sides are optional – select the FK that causes the least amount of nulls.

IF: both sides are mandatory - select the FK that causes the least amount of nulls.

Time-variant data refers to data that change value over time and whose requirements mandate you to keep a history. To maintain the history you need to create an entity containing new values, the date of change, and any time-relevant data. This entity maintains a 1:* relationship with the entity where the history is kept.

Also look at the DATA MODELING checklist on page 233.

Chapter 7 – Normalisation

Definition – Normalisation is the process of evaluating and correcting table structures to minimise data redundancies and thus reducing the likelihood of data abnormalities.
It involves assigning attributes to tables based on the concept of determination.

The following abnormalities could be reduced with normalisation
Update anomalies
Insertion anomalies
Deletion anomalies

The objective is to create tables with the following characteristics:
Each table represent a single subject. E.g. all student info is in one table
No data item will be unnecessary stored in more than one table
All attributes in a table are dependent on the primary key, the entire primary key and nothing but the primary key.

The following steps should be followed to convert a file to 1NF
1. Eliminate the repeating groups
2. Identify the primary key
3. Identify all dependencies (using the dependency diagram)
    - identify your partial and transitive dependencies.
    - partial – a dependency based on only a part of a composite primary key
    - transitive – as dependency from one non-prime attribute to another non-prime

1NF thus – (1) all key attributes are defined, (2) there are no repeating groups, (3) all attributes are dependent on the primary key

The following steps should be followed to convert 1NF to 2NF
1. Write each key component on a separate line
2. Assign corresponding dependent attributes

2NF thus – when in 1NF and (2) include no partial dependencies, that is, no one or more attributes may be functionally dependent on non-key attributes

The following steps should be followed to convert 2NF to 3NF
1. Identify each new determinant
2. Identify the dependent attributes
3. Remove the dependent attributes from transitive dependencies

3NF thus when in 2NF and it contain no transitive dependencies

How to improve your design (page 258)
Evaluate PK assignments
Evaluate naming conventions
Refine attribute atomicity – atomic attribute is one that cannot be further subdivided
Identify new attributes
Identify new relationships

Refine PK as required data granularity – refers to the level of detail represented by the values stored in a table row
Maintain historical accuracy
Evaluate using derived attributes

The Boyce-codd normal form:
A table is in BCNF when every determinant in the table is a cardidate key. When a table thus have only one cardidate key the 3NF and BCNF are the same

Demoralization – in larger number of tables additional I/O operations may impact on processing logic and denormalization must be considered to avoid that.

Chapter 12 – Transaction management and concurrency control

Transaction - is a sequence of database transactions that access the database. A transaction must be a logical unit of work that, that is, no portion of the transaction can exist by itself. Either all parts are executed or the transaction is aborted. A transaction takes a database from one consistent state to another. A consistent database state is one in which all data integrity constrains and satisfied.

A transaction has 5 main properties:
1. Atomicity – all parts of the transaction is executed otherwise the transaction will aboard
2. Consistency – indicates the permanence of the database's consistent state.
3. Isolation – data use by one transaction cannot be used by another until the first transaction is completed
4. Durability – the changes made by the transaction cannot be rolled back once the transaction is committed
5. Serializability – the results of concurrent execution of transaction is the same as that of transaction being executed in a serial order.

Two SQL statements – COMMIT (save to disk) and ROLLBACK (restores to previous state)

Transaction log – keeps track of all changes made to the database and can be use for recovery (ROLLBACK) purposes.
The transaction log stores

1. A record for the beginning of the transaction
2. For each transaction component (SQL statement)
    - The type of operation (update, delete and insert)
    - The names of the objects affected by the transaction
    - The before after values of the fields being updated
    - Pointers to the previous and next transaction log entries

Concurrency control – coordinates the simultaneous execution of transactions. Concurrent execution of transactions can result in the following main problems - Lost updates, uncommitted data and Inconsistent retrievals.

The scheduler is responsible for establishing the order in the concurrent tx operation is being executed. Transaction execution is critical and ensures database integrity in multi database systems. The following controls exist.

- Locking
    - consist of database, table, page, row and field locking
    - 2 types of locks, binary and shared/exclusive locks
    - Binary locks have 2 states 1(locked) and 0 (unlocked)
    - A shared lock is used when a transaction want to read from a database and no other transaction is updating the same data
    - Several shared or read locks can exist for a particular item
    - An exclusive lock is used when a transaction want to update the database and no other locks are held on the data
- Time stamping
    - Methods assigning a unique time stamp to each transaction and schedules the execution of conflicting transaction in time stamp order
    - The schemes are used to decide which transaction is rolled back and which transaction continuous executing – wait/die and wound/wait
- Optimistic methods
    - Assumes that the majority of the database transaction does not conflict and that transaction is executed concurrently using private copies of the data.
    - At commit time the private copies updates the database

Deadlocks – when 2 transactions wait for each other to unlock data. E.g. T1 = access data items X and Y and T2 access data item Y and X. Consequentially T1 and T2 waits indefinitely for each other. Such a deadlock is also known as a deadly embrace.

Concurrency control - Locks

Three ways to control deadlocks:

1. Deadlock prevention – the transaction requesting a new lock is aborted when there is a possibility that the deadlock can occur. When aborted, all changes made are rolled back and locks obtained are released.
2. Deadlock detection – the DBMS periodically tests the database for deadlocks, if a deadlock is found, one of the transactions, the victim is aborted (rolled back and restarted) and the other transaction continues.
3. Deadlock avoidance – The transaction must obtain all locks before its executed. The technique avoids the rolled back of conflicting transactions by requiring that locks are obtained in succession. However the serial lock assignment increases action response times.

Concurrency control – Time stamps

The time stamping approach to scheduling concurrent transactions assigns a global, unique time stamp to each transaction. The time allocated controls the sequence submitted to the DBMS. Time stamps have 2 properties. **Uniqueness** and **monotonicity** - insures that time stamp value always increase.
If two transactions conflict, one is stopped, rolled back, rescheduled and assigned a new time stamp.
Disadvantage – two additional time stamp values is stored in the DB, the last read and last update time. Thus increases memory needs and database processing.

Wait/die schemes
If transaction requesting the lock is younger, it will die (roll back) and rescheduled using the same time stamp. In the wait/die scheme, the older transaction waits and the younger is rolled back and rescheduled.

Wound/wait
If the transaction requesting the lock is older, it will pre-empt (wound) the younger transaction (by rolling back) The younger transaction is rescheduled with the new time stamp.
If the transaction requesting the lock is younger, it will wait for the older to complete and locks are released. In the wound/wait scheme, the older transaction rolls back the younger and reschedules it.

As in both schemes one of the two transactions have to wait, a time-out value is assigned and of a lock is not granted before the time-out expires, the transaction is rolled back.

Concurrency control – optimistic methods

The method is based on the assumption that most database operations does not conflict. It does not require locking or time stamping; instead each transaction is executed until committed. Each transaction moves through 2 or 3 phases of read, validate and write.

During the READ - the transaction reads the DB, makes the update to a private copy of the DB in a temp file which is not accessed by the remaining transactions.

During the validation phase – the transaction is validated to ensure changes will not affect the integrity and consistency of the DB. If the validation is positive, the transaction goes to the write phase, else it's restarted and changes discarded

During the write phase – the changes are permanently applied to the DB.

The approach works well on database systems that require few updates. In a heavily used DBMS environment, the management of deadlocks, their prevention and detection is an important DBMS function. However, the cure is sometimes worse than the disease and it may be necessary to employ database recovery techniques.

Database recovery management

DB recovery restores a database to a previous given state. The recovery technique is based on atomic transaction property – all portions of the transaction must be treated as a single, logical unit of work in which all operations are applied and completed to produce a consistent state.

It is a function of the DBMS and allows the DB administrator to schedule automatic backups.
- Full backup – entire dump of the database
- Differential backup – only last modifications of the DB is compared with the previous
- Transaction log backup – backs up only the transaction log operations that are not reflected in the previous backup copy of the database.

Transaction recovery

Important concepts that affect the recovery process

1. Write-ahead-log protocol – the protocol ensures that the transaction log is always written before the database is updated. In case of a failure, the TL can be used to restore the DB to a consistent state.

2. Redundant transaction logs – most DBMS keep several copies of the TL to ensure a physical disk failure will not impair the DBMS.

3, Database buffers – a buffer is a temp storage area in the primary memory use to speed up disk operations. To improve processing time the DBMS reads from the physical disk and stores a copy in the buffer. Later on all buffers are written to physical disk during a single operation.

4. Database checkpoints – a operation in which the DBMS writes all its updated buffers to disk. While this is happening the DBMS does not execute any other requests. A checkpoint operation is also registered in the transaction log. Checkpoints are scheduled by the DBMS several times per hour and play an important role in transaction recovery.

Steps in the recovery process:

1. Identify the last checkpoint in the transaction log. This is the last time the transaction data were saved to disk
2. For a transaction that started and committed after the last checkpoint, nothing needs to be done
3. Transactions committed before the last checkpoint the DBMS redoes the transaction using the afer values in the transaction log and changes are applied in ascending order.
4. For any transaction that has a rollback operation after the last checkpoint, the DBMS uses the transaction log ROLLBACK using the before values in the transaction log. Changes are applied in reverse order.

Chapter 14 – Distributed Database management Systems (DDBMS)

The disadvantages of a centralized database management system

- Performance, could not cope with growing demand of ad hoc queries over more remote locations
- High cost, maintaining and operating a large central (mainframe) database system
- Reliability problems – created by dependence on a central site

Distributed DBMS

	Advantages	**Disadvantages**
1	Data are located near the greatest demand site	Complexity of management and control – applications must recognise data location and stitch them together
2	Faster data access	Security – the probability of security lapses increase when data are located at more sites
3	Faster data processing	Lack of standards – there are no standard communication protocols at DB level. E.g. TCP/IP

		at network level
4	Improved communications – smaller, faster and better communications between departments and customers	Increased storage requirements
5	Reduced operating costs – more cost effective to add workstations to a network than a mainframe	Increased training costs
6	Less danger of a single point of failure	

Distributed processing and Distributed database

DP – the processing is shared among two or more independent sites connected through a network.

DDB – stores a logically related database over two or more physical independent sites

- Distributed processing does not require DDB's but DDB required DP
- Distributed processing may be based on a single DB located on a single computer
- Both DP and DDB require a network to connect all components or fragments

Characteristics of DDBMS – needs the following to be classified as distributed

- Application interface, interacts with the end user and the DB
- Validation to analyse data requests
- Query optimazation – finds the best way to access
- Transformation – determine which data request is are distributed and which are local
- Mapping – to determine the data location of local and remote fragments
- I/O interface – to read or write data from and to permanent local storage
- Security – to provide data privacy at both local and remote databases
- Backup and recovery – to ensure availability in case of failure
- DB administrator
- Concurrency control – manage simultaneous data access

A sully distributed database management system must perform all of the funcions of a centralised DBMS as follows:

1. Receive an application request
2. Validate and analyse the request
3. Map the request logical to physical components
4. Decompose the request into several disk I/O operations
5. Search for, locate, read and validate date
6. Ensure database consistency
7. Validate the data for conditions specified by the user
8. Present the selected data in the required format

DDBMS Components

- Computer workstations that from the network system. The DDB must be independent
- Network hardware and software components that reside in each workstations
- Communication media that carry data from one workstation to another

The transaction processor (TP), which is the software component that request data receives and processes the applications data request. The TP is also known as a Transaction Manager (TM) or Application processor (AP).

The data processor (DP) stores and retrieves data located at the site. The DP is also known as the DM data manager.

The protocols will determine how the distributed database will:
1. Interface with the network to transport data and commands between DP and AT's
2. Synchronize all data received from DPs and route retrieved data to the appropriate TP's
3. Ensure common database functions in a distributed system.

Levels of data and process distribution

1. Single-site processing, single site data (SPSD) – all processing is done on a single CPU or host computer (mainframe). No processing on end user side. The DBMS is located on the host computer.

2. Multiple-site processing, single-site data (MPSD) – multiple processes run on different computers sharing a single data repository. A network file server runs conventional applications that are accessed through a LAN.

3. Multiple-site processing, Multiple-site Data (MPMD) – describes a fully distributed DBMS with support for multiple data processing and transaction processors. DDBMS are classified as either homogeneous (integrate only one type of centralized DBMS over the network) or heterogeneous (integrate different type over the network).

Distributed database transparency features

Distribution transparency – allows a DDB to be treated as a single logical DB
- Fragmentation-, location- and local mapping transparency

Transaction transparency – allows a transaction to update data at several network sites
Failure transparency – ensures that the system continue to operate in the event of a node failure
Performance transparency - ensures that the system will perform as if it were a centralized DBMS
- The objective of query optimization is to minimize the cost associated with access time, communication cost and CPU cost.

Heterogeneity transparency – allows integration of several different local DBMS under a common global schema.

Data replication:
1. Fully replicated database – stores multiple copies of each database fragment at multiple sites
2. Partially replicated database – stores multiple copies of some database fragments at multiple sites
3. Unreplicated database – no duplication stores each database fragment on a single site.

Client/Sever advantages vs. DDBMS

1. Client/Server solutions are less expensive than mainframe
2. It allows a end-user to use a microcomputer
3. More people have PC skills than mainframe skills

4. The PC is already established in the workplace
5. Various data analysis tools exist to facilitate interaction with many DBMSs that are available on the PC market

Disadvantages of Client/Server
1. The Client/Server architecture creates a more complex environment and difficult to manage
2. In increase in users and processing sites increase the security risks.
3. Increase speed and demand also increase the burden on training and maintenance.

Chapter 16 – Database connectivity and web development

Database connectivity refers to the process through which application programs connect and communicate with data repositories. Also known as database middleware.

5 ways to connect to the data source:
- Native SQL connectivity
- Microsoft open database connectivity (ODBC)
- Data access objects (DAO) and Remote Data Objects (RDO)
- Microsoft Object linking and embedding for database (OLE-DB)
- Microsoft ActiveX Data Objects (ADO.NET)

ODBC – you first need to create a data source name (DSN) for the data source to provide:
1. An ODBC driver
2. A DSN name
3. ODBC driver parameters

Internet databases – the following are web-to-database middleware actions

1. The client browser sends a page request to the web server
2. The web server receives and validates the request
3. The web-to-database middleware reads, validates and executes the script using the database connectivity layer
4. The database server executes the query and passes the result back to the web-to-database middleware
5. The web-to-database middleware compiles the result set, generates a HTML formatted page that includes the results and sends it to the web server
6. The web server returns just the HTML page to the client browser
7. The client browser displays the page on the local computer

XML Applications – extensible markup language is a metalanuage used to represent and manipulate data elements. It's designed to facilitate the exchange of structured documents such as orders and invoices over the internet.

One of the main benefits of XML is that it separates data structure from it's presentation processing.