CSCI 585 Midterm

Thursday, October 19, 2023 8:13 PM

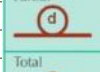
*** CH1:** The file system method was a definite improvement over the manual system but limitations. Data refers to CHARACTERISTICS we select/ define/ specify, for entities. Problems associated with file systems; Extensive programming, Lack of security and limited data sharing, Complex sys administration Uncontrolled data redundancy causes; Poor data security, Data inconsistency, Data-entry errors, Data integrity problems. Data Anomalies; update, insrt, dele Data Modeling is the process of selecting the appropriate STRUCTURE for the given data (eg. hierarchy/tree, network/graph, tables etc). A well -designed database facilitates data management and generates accurate and valuable information. DBMS Database management system. Structural dependence; A data characteristic in which a change in the database schema affects data access, thus requiring changes in all access programs. Structural independence; A data characteristic in which changes in the database schema do not affect data access. Data dependence A data condition in which data representation and manipulation are dependent on the physical data storage characteristics.

*** CH2:** The basic building blocks of all data models are entities, attributes, relationships, and constraints. A constraint is a restriction placed on the data. Constraints are "rules" and help to ensure data integrity. A business rule is a brief, precise, and unambiguous description of a policy, procedure, or principle. Each new data model addresses the shortcomings of previous models. The network model replaced the hierarchical model it made it easier to represent complex (many-to-many) relationships. In turn, the relational model replaced the hierarchical and network models through its simpler data representation, superior data independence, and easy-to-use query language. The OO data model introduced support for complex data within a rich semantic framework. The ERDM added many OO features to the relational model. In recent years, the Big Data phenomenon has stimulated the development of NoSQL databases

*** CH3:** The attribute whose value determines another is called the determinant or the key. A (PK) is an attribute that uniquely identifies any given row. A composite key is composed of more than one attribute, these attributes are known as key attributes. A super key can uniquely identify any row in the table. candidate key is a minimal super key—that is, a super key without any unnecessary attributes. The PK has two requirements; 1 all of the values must be unique, 2 can't be null. SELECT (RESTRICT); Used to yield a set of row. PROJECT; Used to yield a set of cols. UNION; Combines all rows from two tables, the rows have to match and the cols have to have the same values. INTERSECT; only shows rows from both tables, tables need to be union ready for this to work. DIFFERENCE; shows all row not found in the other table, must be union ready to work. PRODUCT; yields all possible pairs of row from two tables. A natural join links tables by selecting from two tables, only those rows that have common (identical) values for common attributes. A 'full outer join' is a union of left outer join and right outer join - output all the rows from both tables, including ones for which there are no matches in the other table

*** CH4 ER:** At the ER modeling level, an entity refers to a table—not to a row—in the relational environment. The ERM refers to a table row as an entity instance or entity occurrence. A compound attribute is not the same as a composite identifier. It's an attribute that can be further subdivided to additional attributes. For example, a phone number may be divided into an area code (615), an exchange number (898), and a four-digit code (2368). The entities that participate in a relationship are also known as participants. Cardinality is the min and max number of entity occurrence associated with 1 occurrence of related entity. Weak vs strong relationship depends on FK. Weak = PK of related entity does not contain PK of the parent entity. Strong = PK of the related entity contains PK of the parent entity. Recursive relationship is a relationship found within a single entity. For example, an EMPLOYEE is 'married' to an EMPLOYEE. The process of database design is iterative. 1 Create a narrative of the organization's description of operations. 2 Identify the business rules based on the description of operations. 3 Identify the main entities and relationships from the business rules. 4 Develop the initial ERD. 5 Identify the attributes and primary keys that describe the entities. 5 Revise and review the ERD. Core Principles of Data modeling; Data Integrity, Normalization, Scalability, Consistency

*** CH5 EER:** A disjoint subtypes, also known as nonoverlapping subtypes, are subtypes that contain a unique subset of the supertype entity set; in other words, each entity instance of the supertype can appear in only one of the subtypes. For example, in Figure 5.2, an employee (supertype) who is a pilot (subtype) can appear only in the PILOT subtype, not in any of the other subtypes. In an ERD, such disjoint subtypes are indicated by the letter d. Overlapping subtypes are subtypes that contain nonunique subsets of the supertype entity set; that is, each entity instance of the supertype may appear in more than one subtype. For example, in a university environment, a person may be an employee, a student, or both. Overlapping subtypes are indicated with the letter o. Specialization is the top-down process of identifying lower-level, more specific entity subtypes from a higher-level entity supertype. Generalization is the bottom-up process of identifying a higher-level, more generic entity supertype from lower-level entity subtypes

TYPE	DISJOINT CONSTRAINT	OVERLAPPING CONSTRAINT
Partial 	Supertype has optional subtypes. Subtype discriminator can be null. Subtype sets are unique.	Supertype has optional subtypes. Subtype discriminators can be null. Subtype sets are not unique. 
Total 	Every supertype occurrence is a member of only one subtype. Subtype discriminator cannot be null. Subtype sets are unique.	Every supertype occurrence is a member of at least one subtype. Subtype discriminators cannot be null. Subtype sets are not unique. 

CROW'S FOOT SYMBOLS	CARDINALITY	COMMENT
	(0,N)	Zero or many; the "many" side is optional.
	(1,N)	One or many; the "many" side is mandatory.
	(1,1)	One and only one; the "1" side is mandatory.
	(0,1)	Zero or one; the "1" side is optional.

*** CH6:** Normalization Process; 1 Each table represents a single subject. For example, a COURSE table will contain only data that directly pertain to courses. 2 No data item will be unnecessarily stored in more than one table (in short, tables have minimum controlled redundancy). The reason for this requirement is to ensure that the data is updated in only one place. 3 All nonprime attributes in a table are dependent on the primary key—the entire primary key and nothing but the primary key. The reason for this requirement is to ensure that the data is uniquely identifiable by a primary key value. 4 Each table is void of insertion, update, or deletion anomalies, which ensures the integrity and consistency of the data. If (A,B) is a primary key, we have partial dependence if (A,B)->(C,D) and B->C [C is only partially dependent on the PK, ie. we only need B to determine C]. In other words, a part of an existing PK is acting like a PK on its own. If X is a primary key, we have a transitive dependency if X->Y and Y->Z [Z is transitively dependent on X, not directly so]. In other words, a non-PK (regular attr) is acting like a PK. 1NF: eliminate repeating groups (partial:y, transitive:y) 2NF: eliminate redundant data (partial:n, transitive:y) 3NF: eliminate fields not dependent on key fields (partial:n, transitive:n). Denormalization may be useful to increase processing speed, and data retrieval but the cost vs benefits need to be weights when undoing some normalization. Data normalization relate to classic software development, it's about minimizing duplication /redundancy. Both utilize modular design, abstraction, encapsulation, both require data integrity, and consistency, both are systematic approaches.

*** CH7:** Data Manipulation; INSERT INTO tablename VALUES(r1,r2,r3..etc); ROLLBACK; DELETE FROM tablename; SELECT * FROM tablename WHERE col = x; Special Operators; BETWEEN 5 AND 10; tuple IS NULL; LIKE 'Smith'; IN (SELECT V_CODE FROM PRODUCT); EXISTS (SELECT * FROM tablename WHERE col = x); ALTER TABLE; ADD - add col; MODIFY - changetype; DROP delete col; ALTER TABLE PRODUCT ADD(P_SELLER CHAR(1)); UPDATE table SET col=2 WHERE col = 3 ORDER BY; SELECT col FROM table WHER col = x ORDER BY Col-list [ASC|DESC]; Aggregate func; COUNT, MIN, MAX, SUM, AVG; GROUP BY needs a agg funct

SELECT V_CODE, (COUNT DISTINCT P_CODE), AVG(P_PRICE) FROM PRODUCT GROUP BY V_CODE HAVING AVG(P_PRICE) < 10; SQL resembles other coding languages in the sense that there are command, built func, operators. But isnt one bec no variables, loops, class def, etc. Entities and classes have datafields but classes can have methods and operations. We can't call methods on data for example price col can't do price.priceOnSale();

* **CH8:** Subqueries and Correlated Queries; A subquery is a query inside another query. Union-Compatible; Num of attri are the same and same datatypes. The UNION operator combines rows from two or more queries without including duplicate rows. VIEWS; CREATE VIEW PRICEisGT50 AS SELECT P_DES, P_QOH, P_PRICE FROM PRODUCT WHERE P_PRICE > 50.00; SQL Func; func can appeare anywhere in a SQL statement where a value or an attri can be used. A Sequence; generates a numeric value in any col in any table. Procedural/SQL is codes that is auto executed by the RDBMS usually invoked by a 'trigger' SELECT P_CODE, P_PRICE FROM PRODUCT WHERE P_PRICE >= (SELECT AVG(P_PRICE) FROM PRODUCT);

* **CH10:** Transaction; Logical unit of work that must be entirely completed or aborted. Transactions have four main properties: Atomicity - all parts of the transaction must be executed; otherwise, transaction aborted. Consistency - the database's consistent state is maintained. Isolation - data used by one transaction cannot be accessed by another until the first one is completed. Durability - changes made by a transaction cannot be rolled back once the transaction is committed. Also, transaction schedules have the property of serializability - the result of the concurrent execution of transactions is the same as that of the transactions being executed in serial order. SQL supports transactions through the use of two statements: COMMIT, which saves changes to disk, and ROLLBACK, which restores the previous database state. Concurrency control coordinates the simultaneous execution of transactions. The execution of transactions can result in three main problems: lost updates, uncommitted data, and inconsistent retrievals. A lock prevents one transaction from using the data item while another transaction is using it. The levels of locks granularity are: database, table, page, row, and field. Two types of locks in database systems: A binary lock can have only two states: locked (1) or unlocked (0). A shared lock is used when a transaction wants to read data and no other transaction is updating that data. Two-phase locking (2PL) defines how transactions acquire and relinquish locks. Two-phase locking guarantees serializability, but it does not prevent deadlocks. The two phases are: 1. A growing phase, in which a transaction acquires all required locks without unlocking any data. Once all locks have been acquired, the transaction is in its locked point. 2. A shrinking phase, in which a transaction releases all locks and cannot obtain a new lock. The two-phase locking protocol is governed by the following rules: a Two transactions cannot have conflicting locks. b No unlock operation can precede a lock operation in the same transaction. c No data is affected until all locks are obtained—that is, until the transaction is in its locked point.

* **CH11:** Performance Tuning; refers to a set of activities and procedures designed to ensure that an end-user query is processed by the DBMS in the least amount of time. End users interact with the DBMS through the use of queries to generate information, using the following sequence: 1. The end-user (client-end) application generates a query. 2. The query is sent to the DBMS (server end). 3. The DBMS (server end) executes the query. 4. The BMS sends the resulting data set to the end-user (client-end) application. DBMSs process queries in three phases. In the parsing phase, the DBMS parses the SQL query and chooses the most efficient access/execution plan. In the execution phase, the DBMS executes the SQL query using the chosen execution plan. In the fetching phase, the DBMS fetches the data and sends the result set back to the client. DBMS performance tuning includes tasks such as managing the DBMS processes in primary memory (allocating memory for caching purposes) and managing the structures in physical storage (allocating space for the data files)

* **CH12:** Centralized Databases; Why Distributed DBMS over Centralized; Performance degradation, High Cost, Reliability, Scalability Organizational Rigidity. What is 2PC protocol; if a portion of a transaction cannot be committed, all changes made at the other site will be undone, for database state consistency. The two phase commit protocol: phase 1: commit request phase, aka 'voting' phase: coordinator sends a 'commit or abort?' (aka 'query to commit' or 'prepare to commit') message to each participating transaction node; each node responds (votes), with a 'can commit' (aka 'ready') or 'need to abort' (aka 'abort') message; phase 2: commit phase, aka 'completion' phase: if in phase 1, all nodes responded with 'can commit', the coordinator sends a 'commit' phase to each node; each node commits, and sends an acknowledgment to the coordinator or, if in phase 1, any node responded with 'need to abort', the coordinator sends an 'abort' message to each node; each node aborts, and sends an acknowledgment to the coordinator; All sorts of variations exist, but the above is the overall (simple, robust) idea. It's an all-or-nothing scheme - either all nodes of a distributed transaction locally commit (in which case the overall transaction is successful), or all of them locally abort so that the DB is not left in an inconsistent state (in which case the overall transaction fails and needs to be redone). In the 'ACID' (Atomicity, Consistency, Isolation, Durability) world of older relational DBs, the CAP Theorem was a reminder that it is usually difficult to achieve C,A,P all at once, and that consistency is more important than availability. In today's 'BASE' (Basically Available, Soft_state, Eventually_consistent) model of non-relational (eg. NoSQL) DBs, we prefer to sacrifice consistency in favor of availability.

* **CH14:** Databases Connectivity; Database connectivity refers to the mechanisms through which application programs connect and communicate with data repositories. Database middleware; Database connectivity software through which application programs connect and communicate with data repositories. ODBC, OLE-DB, and ADO.NET form the backbone of Microsoft's Universal Data Access (UDA) architecture, a collection of technologies used to access any type of data source and manage the data through a common interface. Developed in the early 1990s, Open Database Connectivity (ODBC) is Microsoft's implementation of a superset of the SQL Access Group Call Level Interface (CLI) standard for database access. ODBC is probably the most widely supported database connectivity interface. ODBC allows any Windows application to access relational data sources, using SQL via a standard application programming interface (API). The Webopedia online dictionary (www.webopedia.com) defines an API as "a set of routines, protocols, and tools for building software apps." Database access through the web is achieved through middleware. Database Internet Connectivity, allows new innovative services that permit rapid response bringing new services and products to market. Allows anywhere, anytime data access using mobile smart devices via the internet. Yield fast and effective information dissemination through universal access. Web Application Servers; Middleware apps that expand the functionality of web server by linking them to a wide range of services. Uses include; Connect to query database, Create dynamic web search pages, enforce referential integrity. Exp include WebLogic, WebSphere, JRun

COMMAND OR OPTION	DESCRIPTION
CREATE SCHEMA AUTHORIZATION	Creates a database schema
CREATE TABLE	Creates a new table in the user's database schema
NOT NULL	Ensures that a column will not have null values
UNIQUE	Ensures that a column will not have duplicate values
PRIMARY KEY	Defines a primary key for a table
FOREIGN KEY	Defines a foreign key for a table
DEFAULT	Defines a default value for a column (when no value is given)
CHECK	Validates data in an attribute
CREATE INDEX	Creates an index for a table
CREATE VIEW	Creates a dynamic subset of rows and columns from one or more tables (see Chapter 8, Advanced SQL)
ALTER TABLE	Modifies a table's definition (adds, modifies, or deletes attributes or constraints)
CREATE TABLE AS	Creates a new table based on a query in the user's database schema
DROP TABLE	Permanently deletes a table (and its data)
DROP INDEX	Permanently deletes an index
DROP VIEW	Permanently deletes a view

COMMAND OR OPTION	DESCRIPTION
INSERT	Inserts row(s) into a table
SELECT	Selects attributes from rows in one or more tables or views
WHERE	Restricts the selection of rows based on a conditional expression
GROUP BY	Groups the selected rows based on one or more attributes
HAVING	Restricts the selection of grouped rows based on a condition
ORDER BY	Orders the selected rows based on one or more attributes
UPDATE	Modifies an attribute's values in one or more table's rows
DELETE	Deletes one or more rows from a table
COMMIT	Permanently saves data changes
ROLLBACK	Restores data to their original values

Comparison operators	
=, <, >, <=, >=, <>	Used in conditional expressions
Logical operators	
AND/OR/NOT	Used in conditional expressions
Special operators	
BETWEEN	Checks whether an attribute value is within a range
IS NULL	Checks whether an attribute value is null
LIKE	Checks whether an attribute value matches a given string pattern
IN	Checks whether an attribute value matches any value within a value list
EXISTS	Checks whether a subquery returns any rows
DISTINCT	Limits values to unique values

Aggregate functions	Used with SELECT to return mathematical summaries on columns
COUNT	Returns the number of rows with non-null values for a given column
MIN	Returns the minimum attribute value found in a given column
MAX	Returns the maximum attribute value found in a given column
SUM	Returns the sum of all values for a given column
AVG	Returns the average of all values for a given column