CpSc 4620/6620: Database Management Systems (DBMS) (TEXNH Approach)

Relational Model

James Wang

Relational Model

- Before refining the ER model, it is necessary to understand the relational model.
- The relational model was formally introduced by Dr. E. F. Codd in 1970 and has evolved since then, through a series of revisions.
- The model provides a simple, yet rigorously defined, concept of how users perceive data.
- The relational model represents data in the form of two-dimensional tables.
- Each table represents some real-world person, place, thing, or event about which information is collected.

Relational Database

- A relational database is a collection of two-dimensional relational tables.
- The organization of data into relational tables is known as the logical view of the database.
- A relational table is a flat file composed of a set of named columns and an arbitrary number of unnamed rows.
- The columns of the tables contain information about the table.
- The rows of the table represent occurrences of the "thing" represented by the table.
- A data value is stored in the intersection of a row and column.
- Each named column has a domain, which is the set of values that may appear in that column.

A Simple Bibliographic Database

Author

au_id	au_lname	au_fname	address	city	state
123-456-789	White	John	123 maple way	Clemson	SC
345-234-567	Green	David	345 tiger blvd	Clemson	SC
333-567-987	Dull	Ann	3410 Blonde St.	Berkeley	CA

Title

title_id	title	type	price	pub_id
A1234	PHP Programming	Technical	78.0	2345
B3452	Wall Street Secrets	Business	1000.0	1234
C3648	Is Anger the Enemy?	Psychology	10.99	3566

Publisher

pub_id	pub_name	city
2345	Low tech warehouse	Columbia
1234	Dow Jones	New York
3566	Bob's Publishing	Greenville

Author_Title

au_id	title_id
333-567-987	A1234
123-456-789	B3452
345-234-567	C3648

Properties of Relational Tables

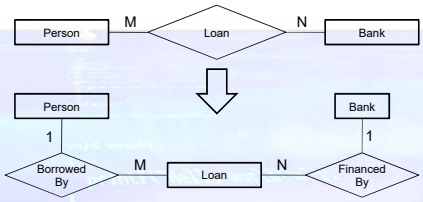
- **Values are atomic:** This property implies that columns in a relational table are not repeating group or arrays.
- **Column values have the same type:** This means that all values in a column come from the same domain.
- **Each row is unique:** This property ensures that no two rows in a relational table are identical; there is at least one column, or set of columns, the values of which uniquely identify each row in the table.
- **The sequence of the column is insignificant:** The ordering of the columns in the relational table has no meaning. Columns can be retrieved in any order and in various sequences.
- **The sequence of the rows is insignificant:** The ordering of the rows in the relational table has no meaning.
- **Each column has a unique name:** In general, a column name need not be unique within an entire database but only within the table to which it belongs.

Relationships and Keys

- A *relationship* is an association between two or more tables. Relationships are expressed in the data values of the primary and foreign keys.
- A *primary key* is a column or columns in a table whose values uniquely identify each row in a table.
- A *foreign key* is a column or columns whose values are the same as the primary key of another table.
- The relationship is made between two relational tables by matching the values of the foreign key in one table with the values of the primary key in another.
- A relationship table is necessary to express the many-to-many relationship between two tables.
- Keys enable tables in the database to be related with each other and allow efficiently navigate the database.

Refining ER Diagram

- **Entities Must Participate In Relationships:** An entity must be related to another entity in the database unless there is only one table in this database.
- **Resolve many-to-many relationship:** Relational model cannot represent many-to-many relationship because the relationships are defined through keys. Therefore, we must convert the many-to-many relationship in ER Diagram into 1-to-many relationship first.



7

Refining ER Diagram (cont.)

- **Transform N-ary Relationships into Binary Relationships:** Complex relationships cannot be directly implemented in the relational model so they should be resolved early in the modeling process. The strategy is to add a new entity to represent the complex relationship. The complex relationship replaced by an association entity and the original entities are related to this new entity.
- **Eliminate redundant relationships:** A redundant relationship is a relationship between two entities that is equivalent in meaning to another relationship between those same two entities that may pass through an intermediate entity.

8

Primary and Foreign Keys

- **Primary and foreign keys are the most basic components on which relational theory is based.**
  - Primary keys enforce entity integrity by uniquely identifying entity instances.
  - Foreign keys enforce referential integrity by completing an association between two entities.
- **Steps for adding primary and foreign keys:**
  - Identify and define the primary key attributes for each entity.
  - Validate primary keys and relationships.
  - Migrate the primary keys to establish foreign keys.

9

Define Primary Key Attributes

- ***Attributes* are data items that describe an entity.**
- **The *primary key* is an attribute or a set of attributes that uniquely identify a specific instance of an entity.**
- **To qualify as a primary key for an entity, an attribute must have the following properties:**
  - it must have a non-null value for each instance of the entity
  - the value must be unique for each instance of an entity
  - the values must not change or become null during the life of each entity instance
- **An entity may have more than one attribute that can serve as a primary key. Any key or minimum set of keys that could be a primary key is called a *candidate key*.**
- **Once candidate keys are identified, choose one, and only one, primary key for each entity. Candidate keys which are not chosen as the primary key are known as alternate keys.**

10

More on Primary Key

- **Composite Keys:** A primary key containing more than one attribute is known as a *composite key*.
- **Artificial Keys:** Artificial keys are permitted when
  - no attribute has all the primary key properties, or
  - the primary key is large and complex.
- **Primary Key Migration:** A dependent entity depends on the existence of another entity for its identification. Therefore it inherits the entire primary key from the parent entity. Every entity within a generalization hierarchy inherits the primary key of the root generic entity.

11

Rules for Primary Key

- **Every entity must have a primary key to identify entity instances. That is, the primary key attribute cannot be optional (i.e., have null values).**
- **The primary key cannot have repeating values. This is called the No Repeat Rule.**
- **Entities with compound primary keys cannot be split into multiple entities with simpler primary keys. This is called the Smallest Key Rule.**
- **Two entities may not have identical primary keys with the exception of entities within generalization hierarchies.**
- **The entire primary key must migrate from parent entities to child entities and from supertype, generic entities, to subtype, category entities.**

12

Foreign Keys

- A *foreign key* is an attribute that defines a relationship between two entities. Every relationship in the model must be supported by a foreign key. Foreign keys are used to maintain the referential integrity and to allow navigating between different entities.
- Foreign key attributes are not considered to be owned by the entities to which they migrate, because they are reflections of attributes in the parent entities.
- If the primary key of a child entity contains all the attributes in a foreign key, the child entity is said to be "identifier dependent" on the parent entity, and the relationship is called an "identifying relationship." If any attributes in a foreign key do not belong to the child's primary key, the child is not identifier dependent on the parent, and the relationship is called "non-identifying."

13

Non-key Attributes

- Non-key attributes can be in only one entity. Unlike key attributes, non-key attributes never migrate from parent to child entities.
- Assign the attributes to the associated entities begins by the modeler and validate the assignment by normalization.
- Rules:
  - In general, entities with the same primary key should be combined into one entity.
  - Place non-key attributes in the parent entity for parent-child relationships as long as it makes sense.
  - If a parent entity has no non-key attributes, combine the parent and child entities.

14

Multi-valued Attributes

- If an attribute is dependent upon the primary key but is multi-valued,
  - We first create a new child entity and migrate the multi-valued attributes into the child entity.
  - If the multi-valued attribute is unique within the new entity, it becomes the primary key. If not, migrate the primary key from the original entity.
- Example:

Project

Proj_ID	Proj_Name	Task_ID	Task_Name
0001	MeTube	001	Analysis
0001	MeTube	002	Design
0001	MeTube	003	Implementation
0002	DB	001	Analysis
0002	DB	003	Implementation

  - Project(proj_ID, Proj_Name, Task_ID)
  - Task(Task_ID, Task_Name)

15

Normalization

- Database normalization is designed to reduce data duplication and ensure the integrity of the database. It simplifies development and maintenance of the database and contributes to its extensibility
- Database tables can be normalized to varying extents. Higher degrees of normalization typically involve more tables and create the need for a larger number of joins, which can reduce performance.
- Normal forms:
  - 1NF (first normal form)
  - 2NF (second normal form)
  - 3NF (third normal form)
  - BCNF (Boyce-Codd normal form)
  - 4NF (fourth normal form)
  - 5NF (fifth normal form)

16

Problems addressed by normalization

- Update Anomaly:
  - The same information may be expressed in multiple records; therefore updates to the table may result in logical inconsistencies.
- Example:
  - Each record in an unnormalized "Employees' Skills" table might contain an Employee ID, Employee Address, and Skill; thus a change of address for a particular employee will potentially need to be applied to multiple records (one for each of his skills).
  - If the update is not carried through successfully—if, that is, the employee's address is updated on some records but not others—then the table is left in an inconsistent state.

17

Problems addressed by normalization (cont.)

- Insert Anomaly:
  - There are circumstances in which certain facts cannot be recorded at all. In the previous example, if it is the case that Employee Address is held only in the "Employees' Skills" table, then we cannot record the address of an employee whose skills are not yet known.
- Delete Anomaly:
  - There are circumstances in which the deletion of data representing certain facts necessitates the deletion of data representing completely different facts.
  - Suppose a table has the attributes Student ID, Course ID, and Lecturer ID (a given student is enrolled in a given course, which is taught by a given lecturer).
  - If in the early stages of enrolment the number of students on the course temporarily drops to zero, then the last of the records referencing that course must be deleted—meaning, as a side-effect, that the table no longer tells us which lecturer has been assigned to teach the course.

18

Definitions

- **Functional dependency:** Attribute B has a functional dependency on attribute A if, for each value of attribute A, there is exactly one value of attribute B.
  - An attribute may be functionally dependent either on a single attribute or on a combination of attributes.
  - **Trivial functional dependency:** An attribute depends on a superset of itself.
  - **Full functional dependency:** An attribute is fully functionally dependent on a set of attributes X if it is a) functionally dependent on X, and b) not functionally dependent on any proper subset of X. For instance, {Employee Address} has a functional dependency on {Employee ID, Skill}, but not a *full* functional dependency, for it is also dependent on {Employee ID}.
  - **Transitive dependency:** A transitive dependency is an indirect functional dependency, one in which $X \rightarrow Z$ only by virtue of $X \rightarrow Y$ and $Y \rightarrow Z$.
  - **Multivalued dependency:** A multivalued dependency is a constraint according to which the presence of certain rows in a table implies the presence of certain other rows.
  - **Join dependency:** A table *T* is subject to a join dependency if *T* can always be recreated by joining multiple tables each having a subset of the attributes of *T*.

19

Definitions (more)

- **Superkey:** A superkey is an attribute or set of attributes that uniquely identifies rows within a table; in other words, two distinct rows are always guaranteed to have distinct superkeys. {Employee ID, Employee Address, Skill} would be a superkey for the "Employees' Skills" table; {Employee ID, Skill} would also be a superkey.
- **Candidate key:** A candidate key is a minimal superkey, that is, a superkey for which we can say that no proper subset of it is also a superkey. {Employee Id, Skill} would be a candidate key for the "Employees' Skills" table.
- **Non-prime attribute**: A non-prime attribute is an attribute that does not occur in any candidate key. Employee Address would be a non-prime attribute in the "Employees' Skills" table.
- **Primary key:** Most DBMSs require a table to be defined as having a single unique key, rather than a number of possible unique keys. A primary key is a candidate key which the database designer has designated for this purpose.

20

First Normal Form (1NF)

- **A relation (table) must not have any duplicate records. In other words, it must have at least one candidate key.**
- **Every column must be atomic, i.e. single-valued with respect to its datatype. In other words, a column may represent exactly one member from its domain.**
  - For example, a date column carrying two dates is a 1NF violation. On the other hand, a datatype may be arbitrarily complex. Therefore, a hypothetical date-range datatype might indeed carry two dates (or rather, one date range) without violating 1NF.
  - Sometimes this second requirement is expressed as "there may not be repeating groups", leading to some prevalent misconceptions. The first misconception is that 1NF precludes a series of columns repeating the same domain. The second misconception is that 1NF does not allow embedded lists.
  - **Note:** 1NF is not well defined through years and will be still ambiguous. (http://en.wikipedia.org/wiki/First_normal_form).

21

Second Normal Form (2NF)

- **The table (relation) must be in 1NF.**
- **Functional dependencies of non-prime attributes on candidate keys are full functional dependencies. If a non-prime attribute of a table is functionally dependent on only a part (subset) of a candidate key, this table is not in 2NF.**
  - For example, in an "Employees' Skills" table with attributes Employee ID, Employee Address, and Skill, the combination of Employee ID and Skill uniquely identifies records within the table. However, Employee Address depends on only Employee ID. Thus, this table is not in 2NF.
- **Note: if every candidate key in a 1NF table contains only one attribute, this table is in 2NF.**
  - http://en.wikipedia.org/wiki/Second_normal_form

22

Third Normal Form (3NF)

- **The table must be in 2NF.**
- **No non-prime attribute is transitively dependent on a candidate key. If a non-prime attribute is only *indirectly* dependent (transitively dependent) on a candidate key, this table is not in 3NF.**
  - For example, consider a "Departments" table with attributes Department ID, Department Name, Manager ID, and Manager Hire Date; and suppose that each manager can manage one or more departments. {Department ID} is a candidate key. Although Manager Hire Date is functionally dependent on the candidate key {Department ID}, this is only because Manager Hire Date depends on Manager ID, which in turn depends on Department ID. This transitive dependency means the table is not in 3NF.
- **Alternative way to determine if a table is in 3NF:**
  - For any functional dependency $X \rightarrow A$, at least one of the following conditions holds:
  - 1) *X* contains *A*, or 2) *X* is a superkey, or 3) *A* is a prime attribute (i.e., *A* is contained within a candidate key)

  (http://en.wikipedia.org/wiki/Third_normal_form)

23

Boyce-Codd Normal Form (BCNF)

- **A table is in Boyce-Codd normal form (BCNF) if and only if, for every one of its non-trivial functional dependencies $X \rightarrow Y$, *X* is a superkey. i.e., *X* is either a candidate key or a superset thereof.**
- **BCNF is slightly more stringent than 3NF. When X is neither a candidate key nor a superset of a candidate key, a table can still be a 3NF as long as Y is a prime attribute. But this case is very rare.**
- **2NF prohibits partial functional dependencies of non-prime attributes on candidate keys.**
- **3NF prohibits transitive functional dependencies of non-prime attributes on candidate keys.**
- **BCNF does not permit any functional dependency in which the determinant set of attributes is not a candidate key (or superset of a candidate key).**

  (http://en.wikipedia.org/wiki/Boyce-Codd_normal_form)

24

Fourth Normal Form (4NF)

- **A table is in fourth normal form (4NF) if and only if, for every one of its non-trivial multivalued dependencies *X* -->> *Y*, *X* is a superkey, i.e., *X* is either a candidate key or a superset thereof.**
- **4NF ensures that independent multivalued facts are correctly and efficiently represented in a database design.**
- **4NF is the next level of normalization after Boyce-Codd normal form (BCNF).**
- **A table with a multivalued dependency often causes redundancy. 4NF removes this redundancy.**

(http://en.wikipedia.org/wiki/Fourth_normal_form)

25

Fifth Normal Form (5NF)

- **The table must be in 4NF.**
- **A 4NF table is said to be in the 5NF if and only if every join dependency in it is implied by the candidate keys.**
- **5NF is designed to reduce redundancy in relational databases recording multi-valued facts by isolating semantically related multiple relationships.**
  - These are situations in which a complex real-world constraint governing the valid combinations of attribute values in the 4NF table is not implicit in the structure of that table.
  - If such a table is not normalized to 5NF, the burden of maintaining the logical consistency of the data within the table must be carried partly by the application responsible for insertions, deletions, and updates to it; and there is a heightened risk that the data within the table will become inconsistent.
  - In contrast, the 5NF design excludes the possibility of such inconsistencies.

(http://en.wikipedia.org/wiki/Fifth_normal_form)

26

Data Integrity

- **Data integrity is to ensure that the data values in the database are correct and consistent.**
- **Data integrity is enforced in the relational model by entity and referential integrity rules.**
  - **Entity integrity:** the value of the primary key must exist, be unique, and cannot be null.
  - **Referential integrity:** every foreign key value must match a primary key value in an associated table.
- **Most DBMS systems also enforce attribute integrity through the use of domain information.**

27

Insert Rules for Referential Integrity

- **Dependent:** permits insertion of child entity instance only if matching parent entity instance already exists.
- **Automatic:** always permits insertion of child entity instance. If matching parent entity instance does not exist, it is created.
- **Nullify:** always permits the insertion of child entity instance. If a matching parent entity instance does not exist, the foreign key in child is set to null.
- **Default:** always permits insertion of child entity instance. If a matching parent entity instance does not exist, the foreign key in the child is set to previously defined value.
- **Customized:** permits the insertion of child entity instance only if certain customized validity constraints are met.
- **No Effect:** always permits the insertion of child entity instance. No matching parent entity instance need exist, and thus no validity checking is done.

28

Delete Rules for Referential Integrity

- **Restrict:** permits deletion of parent entity instance only if there are no matching child entity instances.
- **Cascade:** always permits deletion of a parent entity instance and deletes all matching instances in the child entity.
- **Nullify:** always permits deletion of a parent entity instance. If any matching child entity instances exist, the values of the foreign keys in those instances are set to null.
- **Default:** always permits deletion of a parent entity instance. If any matching child entity instances exist, the value of the foreign keys are set to a predefined default value.
- **Customized:** permits deletion of a parent entity instance only if certain validity constraints are met.
- **No Effect:** always permits deletion of a parent entity instance. No validity checking is done.

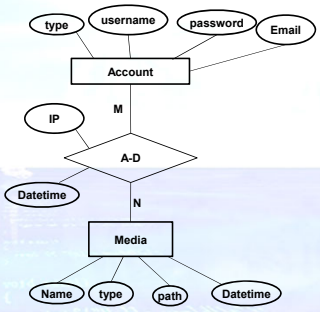
29

What rules should we use?

- **Avoid use of nullify insert or delete rules. Because the parent entity in a parent-child relationship must exist, the nullify insert or delete rule violate this constraint.**
- **Use either automatic or dependent insert rule for generalization hierarchies to ensure that all instances in the subtypes are also in the supertype.**
- **Use the cascade delete rule for generalization hierarchies to enforce that only instances in the supertype can appear in the subtypes.**

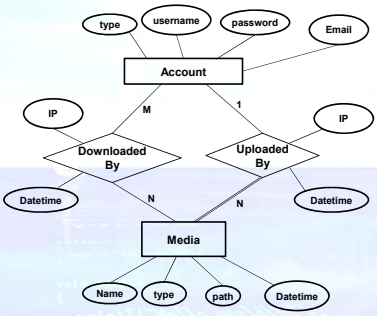
30

What is wrong with the following model?



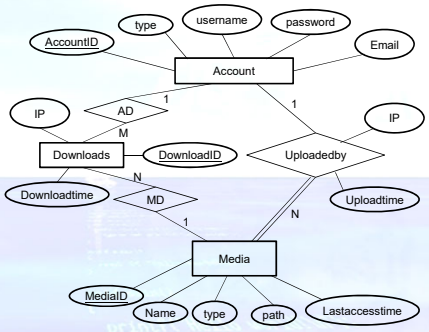
31

How about this?



32

How about this?



33

References

- http://www.youtube.com
- http://archive.comet.ucar.edu/moria/index.jsp
- An Introduction to Database Systems, Eighth Edition, C. J. Date, Addison Wesley, 2004, ISBN: 0-321-19784-4.
- http://en.wikipedia.org/wiki/Database_normalization

34