CTEC323 Lecture 6

Dan Zingaro
OISE/UT

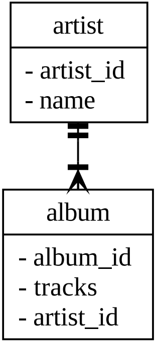
October 9, 2008

Entity-Relationship Model

- Entity-relationship model (ERM) is a conceptual model, independent of the type of database
- It can be used to generate a relational database, hierarchical database, etc. (but we'll be using it for relational databases, of course.)
- Entity-relationship diagrams (ERDs) depict entities, attributes and relationships
- Two notations: Chen and Crow's Foot; we have been and will continue using the latter

Entities and Attributes

- Entity in an ERM corresponds to a table (entity set) in the relational model
- Represent entity by rectangle containing its name
- Can place attributes below the entity name



Types of Attributes

- In an ERD, we can underline attributes that are part of a PK
- Composite attributes can be subdivided to yield additional attributes
  - Address can be divided into street, city, province, etc.
- Simple attributes cannot be subdivided
- Usually appropriate to change composite attributes into a series of simple attributes
- Single-valued attribute can have only a single value (e.g. social insurance number)
- Multivalued attributes can have many values (e.g. people can have more than one phone number)

Multivalued Attributes

- Multivalued attributes should not be implemented in the RDBMS
- There are two things we can do
- We can Create several new attributes, one for each of the multivalued attribute's components
  - But, for something like phone numbers, how do we know how many to include?
  - There will be many nulls since not all components will be applicable to everyone
- Or, we can create a new entity composed of the multivalued attribute's components
  - There is then a 1:M relationship between the original entity and the new one representing the multivalued attribute
  - Bonus: we can add new components without modifying table structure

Derived Attributes

- A derived attribute's value can be calculated from other attributes and so may not actually be stored in the database
- e.g. if we store a person's date of birth, we can calculate their age (age is a derived attribute)
- If we include them, we save CPU processing but must ensure that the data remain current
- If we leave them out, we save storage space but must write more complicated queries
- Storing derived attributes also lets us maintain historical data

Connectivity and Cardinality

- The term connectivity is used to describe relationship classification (i.e. 1:1, 1:M, M:M)
- Cardinality expresses the minimum and maximum number of entity occurrences associated with one occurrence of the related entity
- e.g. assume there is a 1:M relationship between professors and classes. We can put (1, 4) on the class side to indicate that a professor teaches between 1 and 4 classes. We can put (1, 1) on the professor side to indicate that a class is taught by exactly one professor
- Precise cardinality constraints often cannot be enforced by the RDBMS without triggers or support from application programs

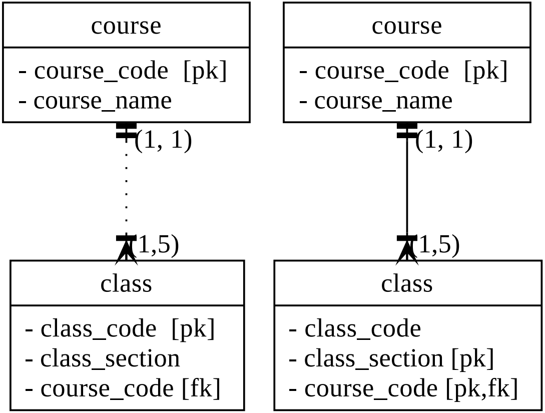
Existence-Dependence

- An entity is existence-dependent if it can exist in the database only when it is associated with another related entity occurrence
- In implementation terms, an entity is existance-dependent if it has a mandatory foreign key (i.e. it cannot be null)
- Exercise: consider the “professor teaches class” relationship. Is class existence-dependent on professor?

Weak and Strong Relationships

- Weak relationship
  - PK of child entity does not contain PK component of the parent entity
  - Indicated in crow's foot notation with a dashed relationship line
- Strong relationship
  - PK of child entity contains a PK component of the parent entity

Weak and Strong Relationships...



(a) Weak Relationship (b) Strong Relationship

Figure: Weak relationship when class' PK is class_code; strong when it is course_code + class_section

Weak and Strong Entities

- A weak entity meets two conditions
  - It is existence-dependent, and
  - It has a PK that is partially or totally derived from the PK of the parent entity

Participation

- Optional participation means that one entity occurrence does not require a corresponding entity occurrence in a particular relationship
- Optional relationship between entities is indicated by a circle on the side of the optional entity
- Existance of optionality indicates that minimum cardinality is 0 for the optional entity
- Existance of mandatory relationship indicates that the minimum cardinality is 1 for the mandatory entity

Degree

- Degree indicates the number of entities associated with a relationship
- Unary: association maintained within a single entity
- Binary: association between two different entities
- Recursive relationship: occurs when a relationship exists between occurrences of the same entity set

Implementing Recursive Relationships

- Consider the recursive (unary) relationship “employee is married to employee”
- One way to implement this is with a single table whose attributes are employee number, employee name, and employee spouse
- Another possibility uses one table with attributes for the employee number and name, and a second table with attributes for employee number and spouse
- A third possibility uses three tables
  - Table with attributes for employee number and name
  - Table with attributes for marriage number and date
  - Table with attributes for marriage number and employee number

Comparison of Implementations

- First implementation
    - Results in storing nulls for employees that are not married to another employee in the company
    - Can yield data anomalies (e.g. when two people divorce, and we only update one employee's row)
- Second implementation
    - Eliminates nulls associated with employees who are not married to another employee in the company
    - We can still record each marriage twice (and introduce inconsistencies by doing so)
- Third implementation
    - Still must be careful (e.g. must have a unique index on the employee number in the third table)
    - What if we have more than two employees with the same marriage number in the third table?

Conflicting Design Goals

- Often must make compromises triggered by conflicting goals of
    - Adherence to design standards (minimize data redundancy)
    - Processing speed (may combine tables to avoid relationships, decreasing data access time because less joins are involved)
    - Information requirements (may expand number of entities and attributes)
- They are in conflict because, for example, when we combine tables for the sake of efficiency, we may no longer be following design standards