



Database Design



Overview: Design Process

- A process of determining the tables needed for a given collection of fields and placing each field in the correct table
- Steps
 1. Identify all required columns
 - from source documents, reports, ideas from current users
 2. Identify entities
 - generally, each entity is placed in its own table
 3. Consider relationships between entities
 4. Normalization process
 5. Produce an ER diagram
 6. Use SQL to create the tables & inter-table relationships

2



1:Many Relationships

- Each record in Table A can have **many** (zero or more) matching records in Table B but each record in Table B has **at most one** matching record in Table A
- Implementation
 - have the primary key of the **one** table appear as a foreign key in the **many** table
 - eg: each **Course** can have many (zero or more) **Sections** but each **Section** is for **one Course**
 - each **ZipCode** can have many **Students** but each **Student** lives in only one **ZipCode**
 - eg: each **Department** can have many **Employees** but each **Employee** works in only one **Department**

3



1:1 Relationships

- Each row in one table has **at most** one matching row in the other table
- A relationship between the primary keys of two tables
- Commonly used to:
 - control field access
 - avoid **nulls**

4



Many:Many Relationships

- Each record in Table A can have **many** matching records in table B and each record in Table B can have **many** matching records in Table A.
- Examples
 - each **Student** can enroll in many **Sections** & each **Section** can enroll many **Students**
 - each **Pitcher** can pitch in many **Games** & each **Game** can use many **Pitchers**
- Implementation:
 - create a third table (composite entity, intersection table, junction table) which includes the primary keys from table A and table B as foreign keys
 - consider using them as a composite primary key in the new table
 - doing so breaks the Many:Many relationship into two separate 1:Many relationships

5



Functional Dependency

- X → Y**
 - "the value in column X functionally determines the value for column Y"
- Examples:

ArticleNum → Title	ArticleNum → Length
ZipCode → City	ZipCode → State
?????? → Final_Grade	
- Whenever 2 of a table's rows have the same X value, they will also have the same Y value
 - i.e., **data redundancy** will occur
- Determinant (X)
 - a column (or collection of columns) which determines value of another
- Dependent (Y)
 - a column whose value is determined by another column

6

TABLE(A, B, C, D, E)

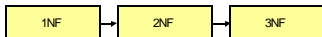
Functional Dependency

- Good Dependency
 - when a column is dependent only on the table's primary key (and on any candidate keys)
- Partial Dependency
 - when a column is dependent on only part of the table's primary key
 - can only occur when a table has a composite primary key
 - if allowed to remain, will cause data redundancy
- Transitive Dependency
 - when one column is dependent on another column that is not the table's primary key
 - if allowed to remain, will cause data redundancy

7

Normalization Process

- Start with an initial set of tables, then apply rules to produce a final set of problem-free tables
- A process that eliminates data redundancy
- Relies on identifying **functional dependencies** of each column



8



First Normal Form

- A table that does not contain **repeating groups**
 - each field must be **atomic** (contain a single item)
- Fix repeating groups by
 - removing the repeating group column(s) and placing them in a table that includes the original table's primary key
- Figure 1.8, pg 15
 - what if there could be 4 locations? 5? 6? ...
 - a table design change is required and potential for null values
- Figure 1.9, pg 15
 - what if there are 4 locations?
 - what if there are 400 locations?

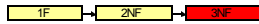
9



Second Normal Form

- A table in 1NF where every nonkey column is dependent on the entire primary key, not part of it
 - i.e., no **partial dependencies**
 - a table in 1NF is automatically in 2NF if its primary key is not a composite
- Fix partial dependencies by
 - identify the functional dependencies for each column
 - place each column in a table where it is functionally dependent on the entire primary key
- Figure 1.10, pg 16
 - what is PHONE_NUMBER dependent on?
 - why is PHONE_NUMBER a problem in this table?

10



Third Normal Form

- A table in 2NF in which every determinant is also a candidate key
 - i.e., no **transitive dependencies**
- Each nonkey column must be mutually independent and dependent only on candidate keys
- Fix transitive dependencies by
 - removing any column that depends on a non-candidate-key
 - place it in a table with the determinant as the primary key and keeping the new table's primary key in the original table as a foreign key
- Figure 1.11, pg 16
 - what is PUBLISHER_PHONE_NUMBER dependent on?

11

Design Process

- Preliminaries
 - 1 Read the case and identify obvious **entities**
 - 2 Begin an ER diagram
 - 3 Re-read the case looking for Many-to-Many relationships
 - 4 Tentatively place each field in a table on the ER diagram
- Normalization
 - 5 Obtain First Normal Form by eliminating **repeating groups**
 - 6 Obtain Second Normal Form by eliminating **partial dependencies**
 - 7 Obtain Third Normal Form by eliminating **transitive dependencies**
- Documentation
 - 8 ER Diagram

12
