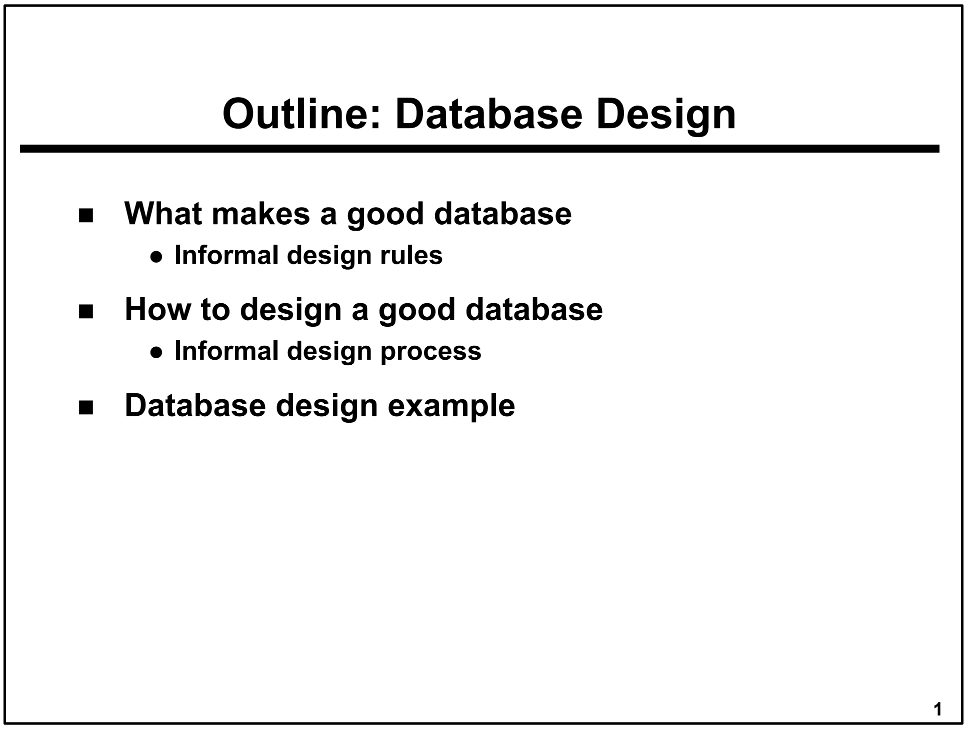
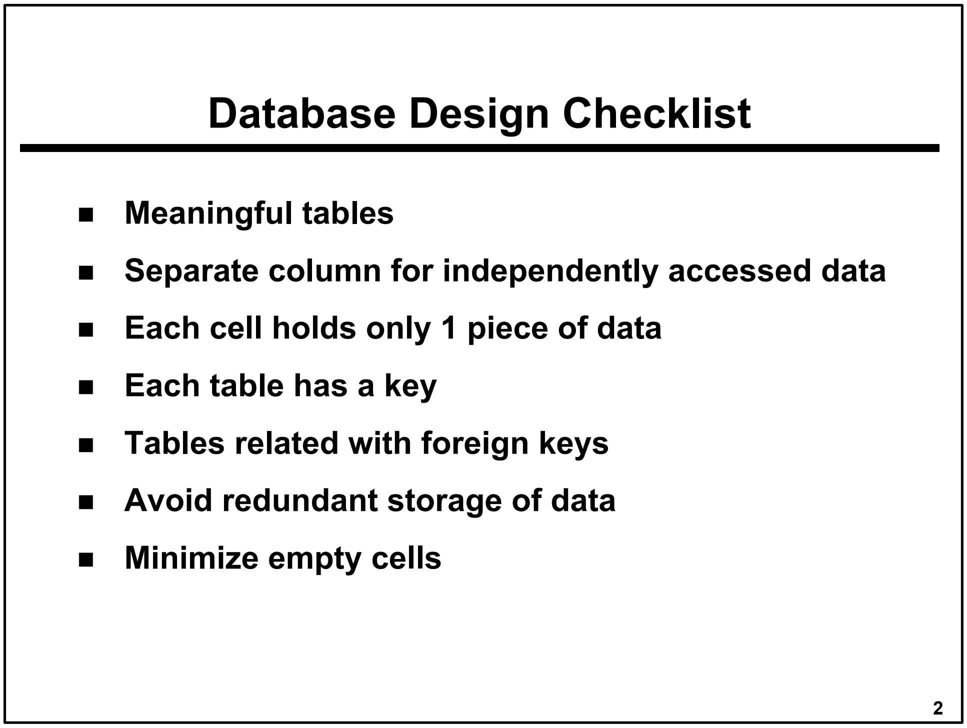


Outline: Database Design

- What makes a good database
  - Informal design rules
- How to design a good database
  - Informal design process
- Database design example

1



Database Design Checklist

- Meaningful tables
- Separate column for independently accessed data
- Each cell holds only 1 piece of data
- Each table has a key
- Tables related with foreign keys
- Avoid redundant storage of data
- Minimize empty cells

2

Meaningful Tables

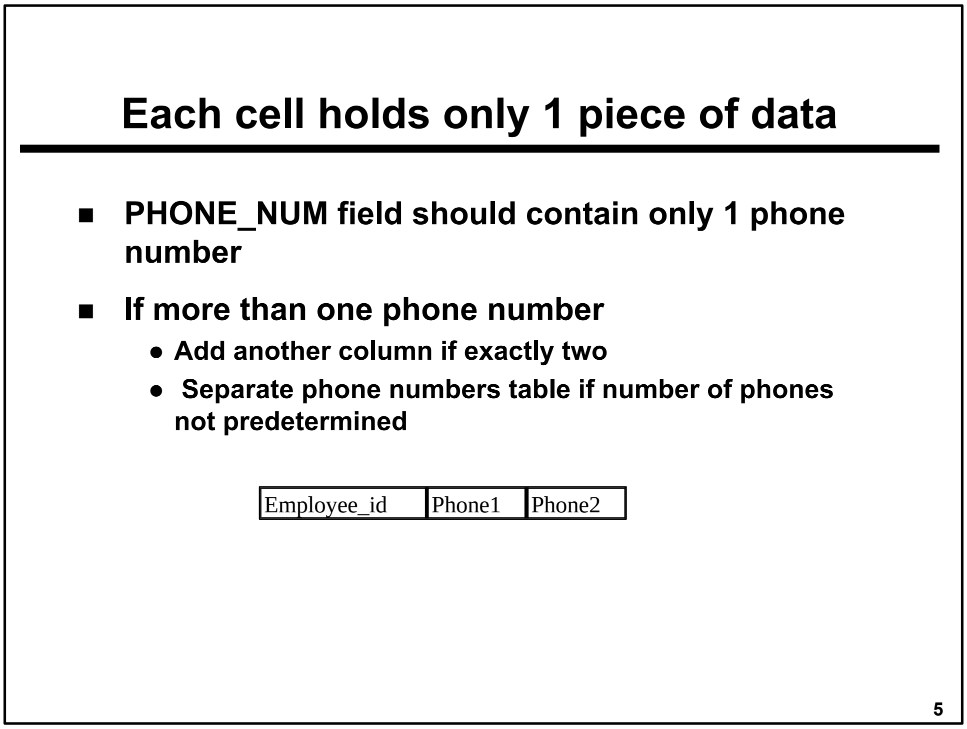
- Each row should represent one instance of an entity or relationship
  - One employee
  - One project-employee relationship
- One table should not contain data about several entities
  - E.g., employee id and department location in separate tables
    - Even though employee is currently assigned to a department, which has a location
    - Easier to update if employee switches departments
- Litmus test: succinct answer to:
  - “What’s in this table?”

Separate column for independently accessed data

- If you'll ever access just part of a column, separate it
- Example: address database
  - May want to sort by zip code
    - Save on postage
    - Targeted mailings
  - Make zip code a separate field instead of one field per line of address

Name	Address_1	Address_2	Address_3

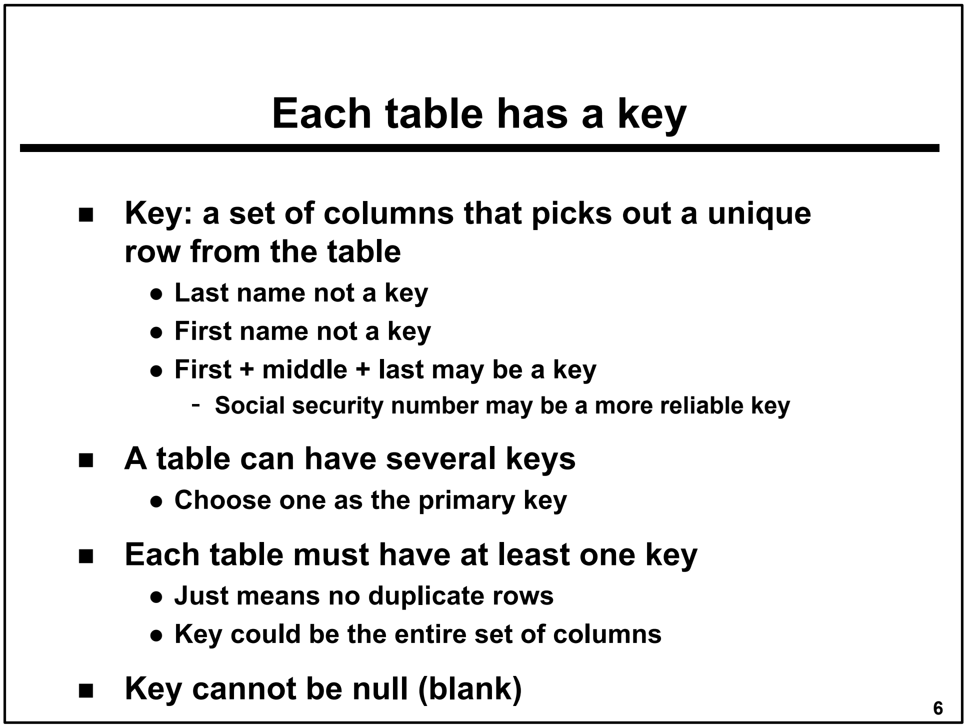
Lname	Fname	Org	Street	City	State	Zip	Country



Each cell holds only 1 piece of data

- PHONE_NUM field should contain only 1 phone number
- If more than one phone number
  - Add another column if exactly two
  - Separate phone numbers table if number of phones not predetermined

Employee_id	Phone1	Phone2



Each table has a key

- Key: a set of columns that picks out a unique row from the table
  - Last name not a key
  - First name not a key
  - First + middle + last may be a key
    - Social security number may be a more reliable key
- A table can have several keys
  - Choose one as the primary key
- Each table must have at least one key
  - Just means no duplicate rows
  - Key could be the entire set of columns
- Key cannot be null (blank)

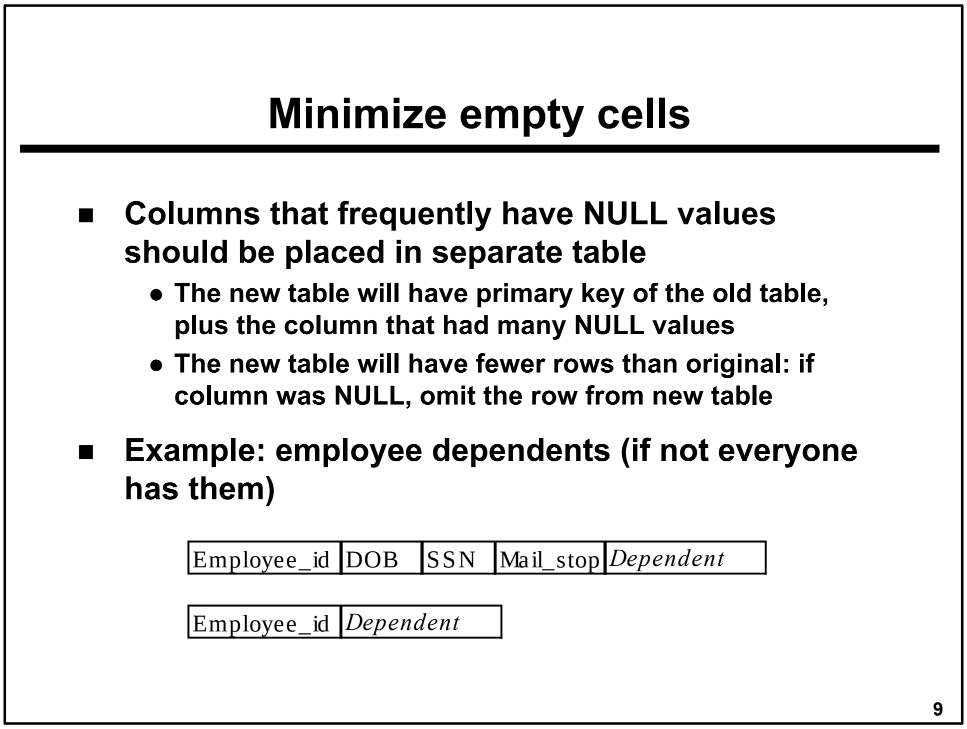
Tables related with foreign keys

- Tables can be related via column(s) in common
- Design goal
  - A row in one table that refers to another table must refer to an existing row in that table
  - Example: Employee table and Department table
    - Don't assign employee to department 10 if that department doesn't exist in other table
  - Foreign key design rule ensures that
- A set of columns in table 1 is a foreign key for table 2 if:
  - The foreign key takes on values from the same domain as the primary key of table 2
  - When the value of the foreign key in table 1 is not null, there is a row in table 2 that has that value

Avoid redundant storage of data

- Redundant storage is wasteful
- Example
  - Suppose employee table keeps track of department and its address for each employee
  - Address repeated for every employee in department
  - What can go wrong?
    - insert new employee
    - modify department address
    - delete last employee for department

Employee_id	Dept_id	Dept_address

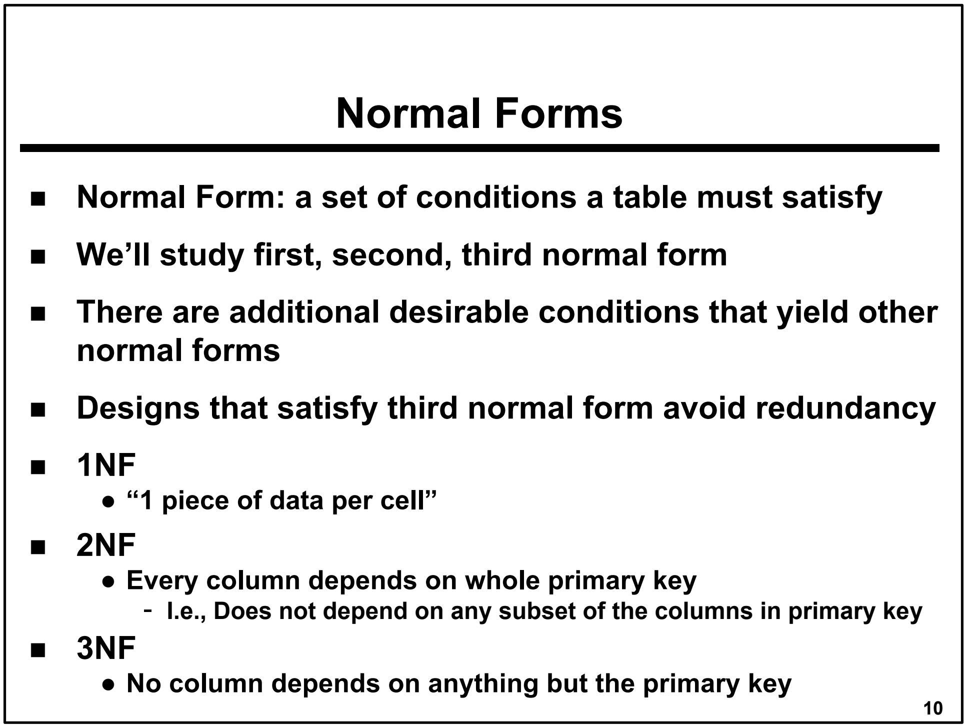


Minimize empty cells

- Columns that frequently have NULL values should be placed in separate table
  - The new table will have primary key of the old table, plus the column that had many NULL values
  - The new table will have fewer rows than original: if column was NULL, omit the row from new table
- Example: employee dependents (if not everyone has them)

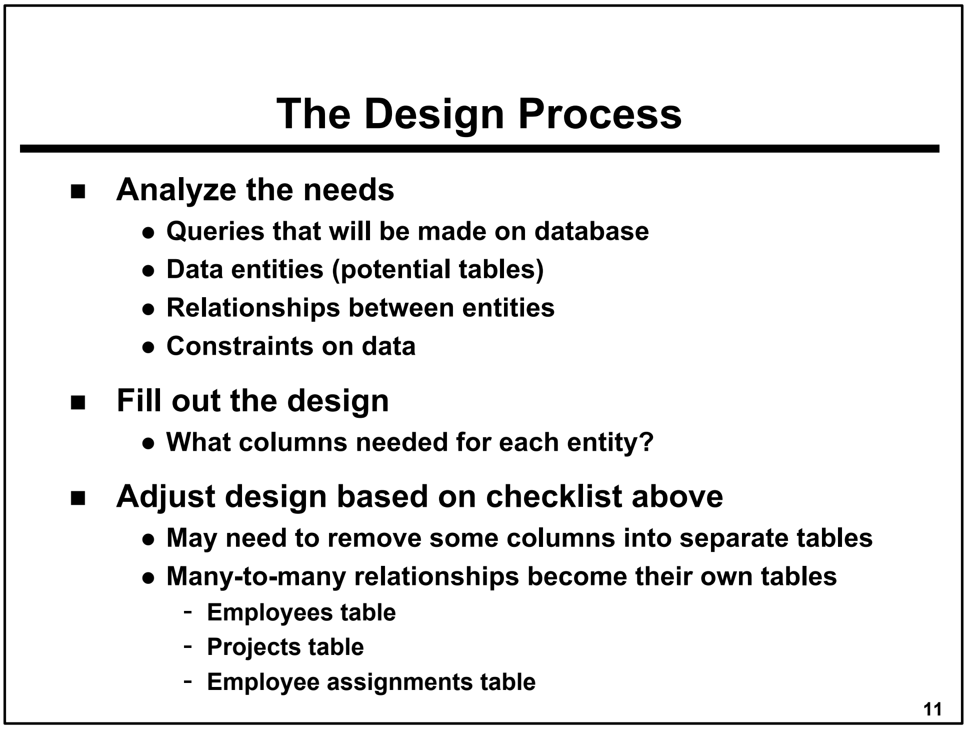
Employee_id	DOB	SSN	Mail_stop	*Dependent*

Employee_id	*Dependent*



Normal Forms

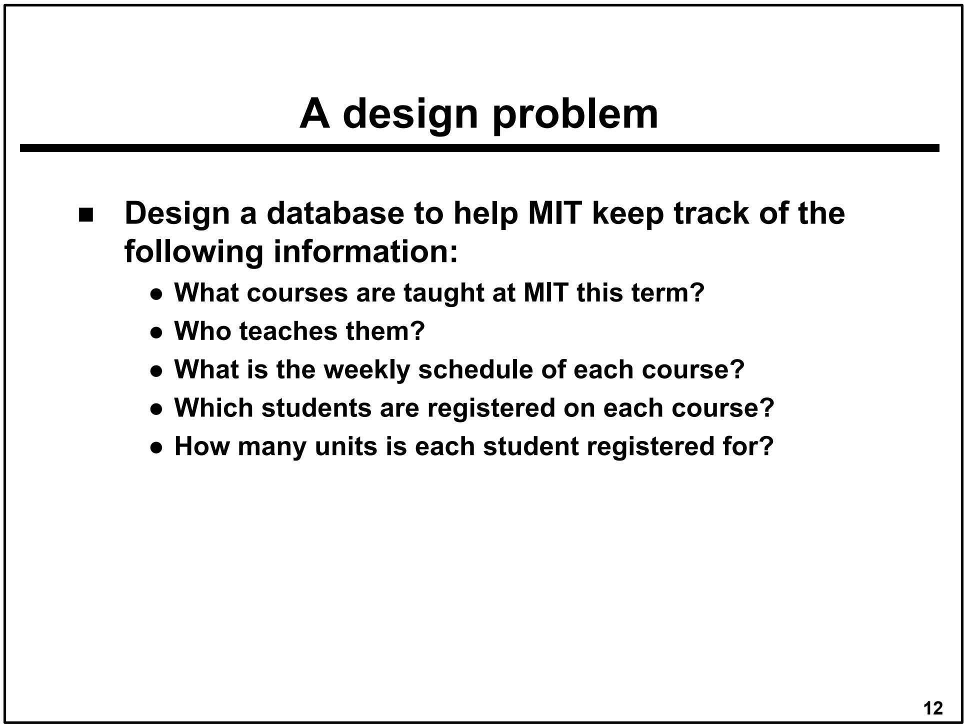
- Normal Form: a set of conditions a table must satisfy
- We'll study first, second, third normal form
- There are additional desirable conditions that yield other normal forms
- Designs that satisfy third normal form avoid redundancy
- 1NF
  - "1 piece of data per cell"
- 2NF
  - Every column depends on whole primary key
    - I.e., Does not depend on any subset of the columns in primary key
- 3NF
  - No column depends on anything but the primary key



The Design Process

- Analyze the needs
  - Queries that will be made on database
  - Data entities (potential tables)
  - Relationships between entities
  - Constraints on data
- Fill out the design
  - What columns needed for each entity?
- Adjust design based on checklist above
  - May need to remove some columns into separate tables
  - Many-to-many relationships become their own tables
    - Employees table
    - Projects table
    - Employee assignments table

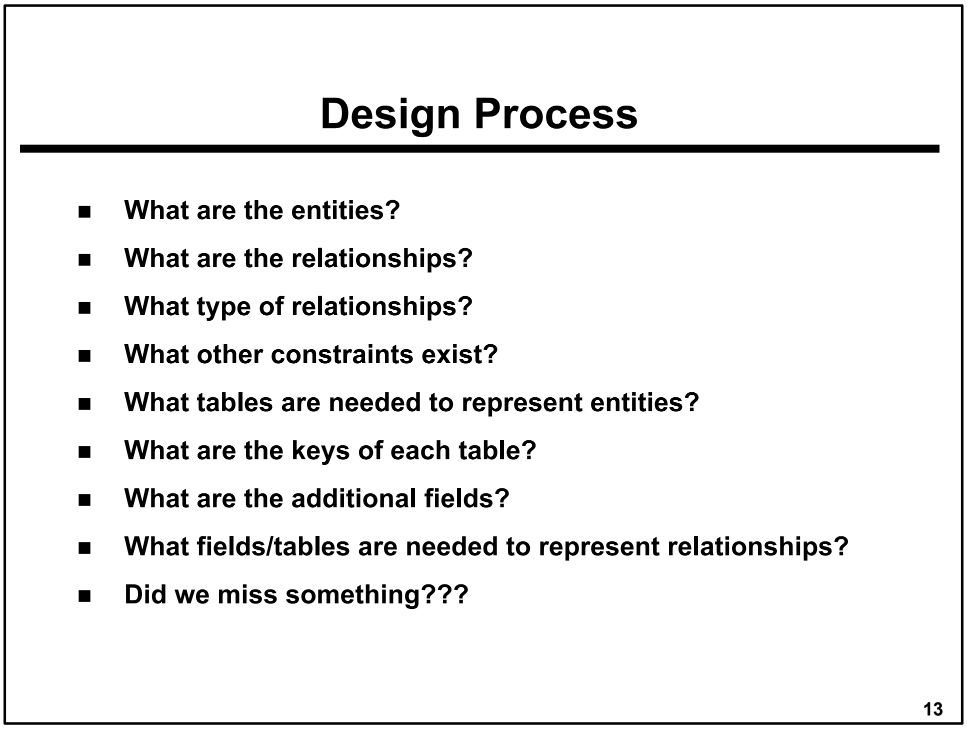
11



A design problem

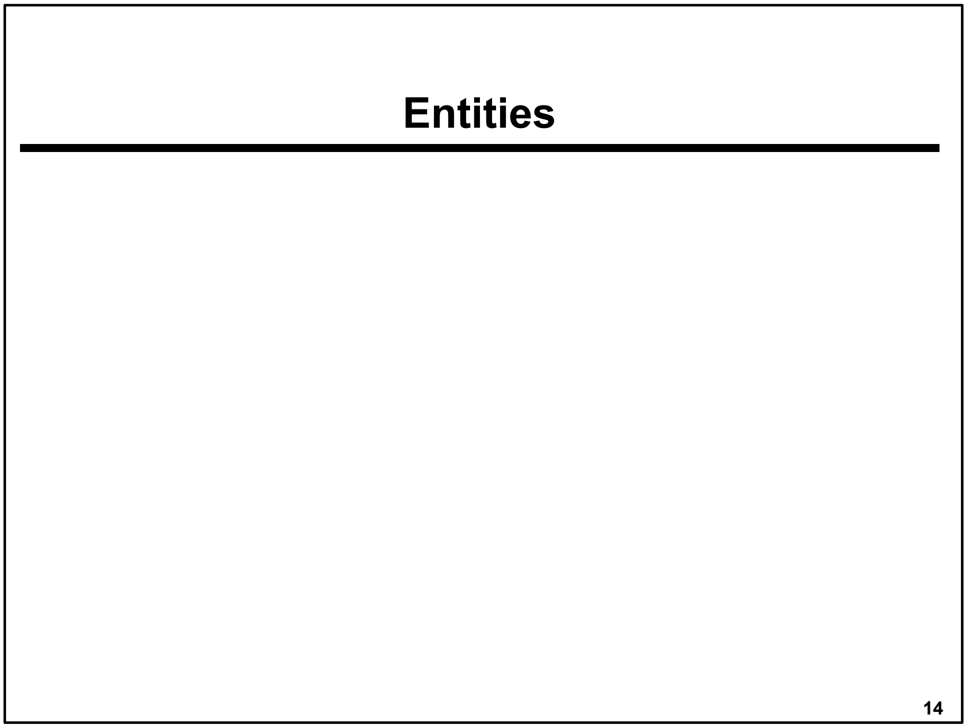
- Design a database to help MIT keep track of the following information:
  - What courses are taught at MIT this term?
  - Who teaches them?
  - What is the weekly schedule of each course?
  - Which students are registered on each course?
  - How many units is each student registered for?

12

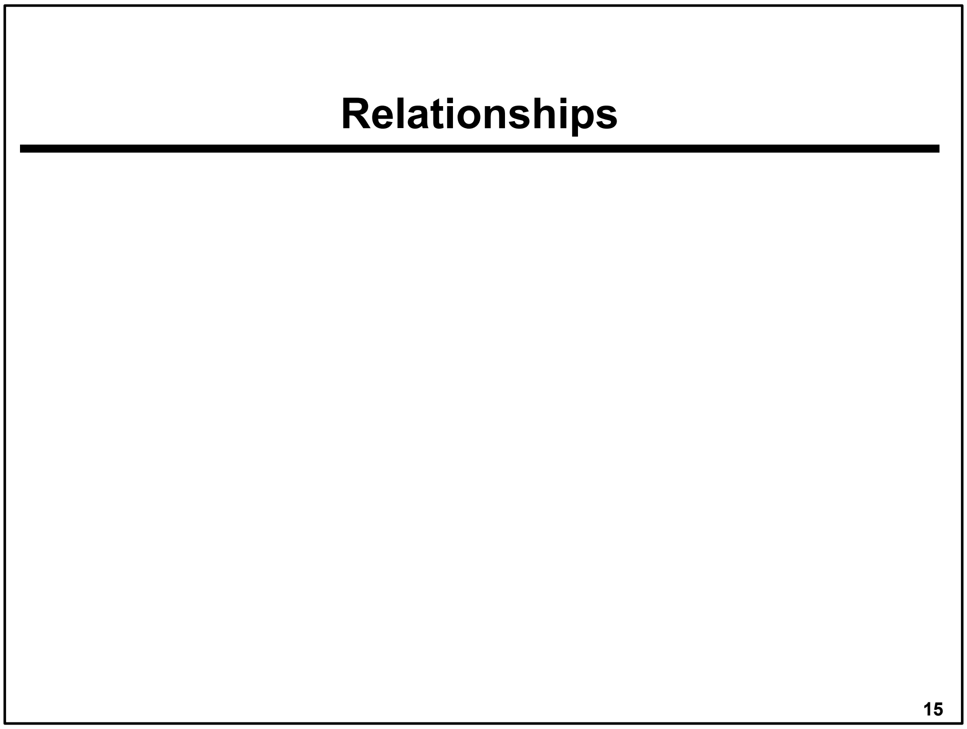


Design Process

- What are the entities?
- What are the relationships?
- What type of relationships?
- What other constraints exist?
- What tables are needed to represent entities?
- What are the keys of each table?
- What are the additional fields?
- What fields/tables are needed to represent relationships?
- Did we miss something???

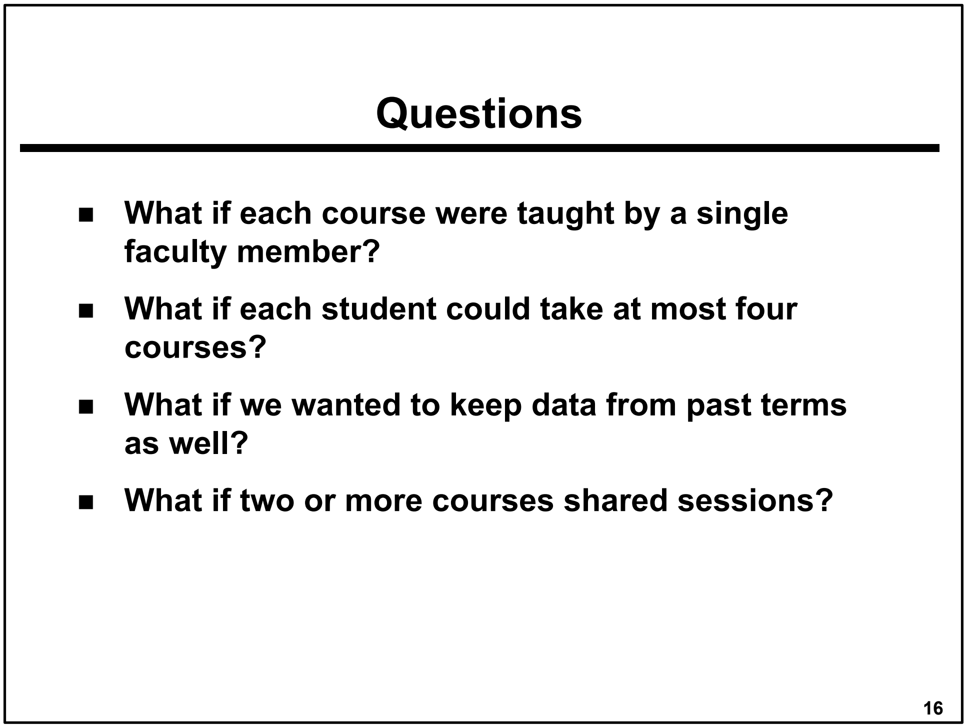


Entities



Relationships

15



Questions

- What if each course were taught by a single faculty member?
- What if each student could take at most four courses?
- What if we wanted to keep data from past terms as well?
- What if two or more courses shared sessions?

16