Exercise

• Design a schema to store the data used by iTunes (or a similar application)
• Things to keep in mind:
  – Can all necessary information be represented?
  – Are properties represented in more than one place?
  – Are common operations efficient?
iTunes information

• Database includes: songs, artists, albums, album artwork, playlists

• Sample fields include: song name, artist, year, albumName, trackNum, genre, lastPlayedDate, playCount, albumArtwork, songRating, albumRating?, albumReleaseDate, artistDOB

• playLists: an ordered list of songs
Relational Schema Design

Conceptual Design

ER Model

Logical design

Relational Schema plus Integrity Constraints

Schema Refinement

Normalized schema
Entity / Relationship Diagrams

Entity sets

Attributes

Relationships

Product

address

buys
Keys in E/R Diagrams

• Every entity set must have a key

![Diagram of a product entity set with attributes name, category, and price]
Multiplicity of E/R Relations

• one-one:

• many-one

• many-many
“A product is made by at most one company”
Referential Integrity Constraints

Each product made by at most one company.
(Some products made by no company)

Each product made by exactly one company.
Multi-way Relationships

Product

Purchase

Person

Store
Q: what does the arrow mean?

A: if I know the store, person, invoice, I know the movie too
Arrows in Multiway Relationships

Q: what do these arrow mean?

A: store, person, invoice determines movie and store, invoice, movie determines person
Roles in Relationships

What if we need an entity set twice in one relationship?
Attributes on Relationships

Product

Purchase

date

Person

Store
Subclasses

Product

- name
- category
- price

isa

Software Product

isa

Educational Product

platforms

Age Group
Design Principles

What’s wrong?

Product → Purchase → Person
Design Principles: What’s Wrong?

Moral: pick the right kind of entities.
Design Principles: What’s Wrong?

Moral: don’t complicate life unnecessarily.
From E/R Diagrams to a Relational Schema

- Entity set $\rightarrow$ relation
- Relationship $\rightarrow$ relation
**Entity Set to Relation**

Product

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>gizmo</td>
<td>gadgets</td>
<td>$19.99</td>
</tr>
</tbody>
</table>
Relationships to Relations

Makes\( (\text{product-name}, \text{product-category}, \text{company-name}, \text{year}) \)

(watch out for attribute name conflicts)
No need for Makes. Modify Product:

Product(name, category, price, startYear, companyName)
Multi-way Relationships to Relations

Product
- name
- price

Purchase
- prodName
- stName
- ssn

Person
- name
- ssn

Store
- name
- address

Purchase(prodName, stName, ssn)
Normalization
Evils of Redundancy

• When a database schema is poorly designed we get anomalies.

• Redundancy is at the root of several problems associated with relational schemas:

  **Redundant storage**: data is repeated
  **Update anomalies**: need to change in several places
  **Insertion anomalies**: may not be able to add data we want to
  **Deletion anomalies**: may lose data when we don’t want to
Anomalies

Hourly_emps( ssn, name, lot, rating, hourly_wages, hours_worked)

Suppose hourly wages is determined by rating.

testing → hourly_wages

<table>
<thead>
<tr>
<th>ssn</th>
<th>name</th>
<th>lot</th>
<th>rating</th>
<th>hourly_wages</th>
<th>hours_worked</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>brutus</td>
<td>48</td>
<td>8</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>85</td>
<td>art</td>
<td>22</td>
<td>8</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>95</td>
<td>bob</td>
<td>35</td>
<td>5</td>
<td>7</td>
<td>30</td>
</tr>
<tr>
<td>96</td>
<td>frodo</td>
<td>35</td>
<td>5</td>
<td>7</td>
<td>32</td>
</tr>
<tr>
<td>22</td>
<td>dustin</td>
<td>35</td>
<td>8</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

• Redundant storage: association between rating 8 and hourly wages 10 repeated 3 times.
• Update anomalies: hourly_wages updated in first tuple but not second
• Insertion anomalies: must know hourly_wage for rating value
• Deletion anomalies: delete all tuples with certain rating value, lost assoc.
Can null values fix problems?

• Not really.
• Insertion anomaly:
  – What if we know rating and hourly_wages for some rating, but there is no employee with that rating?
  – No. ssn can’t be null.
• Deletion anomaly:
  – If last employee with some rating and hourly_wages value is deleted, replace with nulls?
  – No. ssn can’t be null.
Schema Refinement

• Integrity constraints, in particular *functional dependencies*, can be used to identify schemas with such problems and to suggest refinements.
• Main refinement technique: *decomposition* (replacing ABCD with, say, AB and BCD, or ACD and ABD).
• Decomposition should be used judiciously:
  ▪ Is there reason to decompose a relation?
  ▪ What problems (if any) does the decomposition cause?
Data Anomalies

Persons may have several phones:

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

Anomalies:

• Redundancy = repeat data
• Update anomalies = Fred moves to “Bellevue”
• Deletion anomalies = Joe deletes his phone number: what is his city?

SSN → Name, City but not SSN → PhoneNumber
Relation Decomposition

Break the relation into two:

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

Anomalies have gone:
- No more repeated data
- Easy to move Fred to “Bellevue” (how ?)
- Easy to delete all Joe’s phone number (how ?)
Decompositions in General

\[ R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p) \]

\[ R_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \]
\[ R_2(A_1, \ldots, A_n, C_1, \ldots, C_p) \]

\[ R_1 = \text{projection of } R \text{ on } A_1, \ldots, A_n, B_1, \ldots, B_m \]
\[ R_2 = \text{projection of } R \text{ on } A_1, \ldots, A_n, C_1, \ldots, C_p \]
Lossless Decomposition

- Sometimes it is correct:

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
</tr>
</tbody>
</table>
Lossy Decomposition

- Sometimes it is not:

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

What’s wrong??

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>