modification anomalies

<table>
<thead>
<tr>
<th>reviewer</th>
<th>subject</th>
<th>rating</th>
<th>email</th>
<th>ratingstars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chloe Closure</td>
<td>Lucid</td>
<td>3</td>
<td>cc@mit</td>
<td>***</td>
</tr>
<tr>
<td>Chloe Closure</td>
<td>Clover</td>
<td>5</td>
<td>cc@mit</td>
<td>*****</td>
</tr>
<tr>
<td>Ann Alert</td>
<td>Clover</td>
<td>5</td>
<td>aa@mit</td>
<td>*****</td>
</tr>
<tr>
<td>Ben Bitdiddle</td>
<td>Cosi</td>
<td>3</td>
<td>ben@mit</td>
<td>***</td>
</tr>
<tr>
<td>Ben Bitdiddle</td>
<td>Lucid</td>
<td>4</td>
<td>ben@mit</td>
<td>****</td>
</tr>
</tbody>
</table>

again, suppose review database is just one table
› what bad things might happen on update or delete?

deletion anomaly
› delete Ann’s review, and she’s gone too!

update anomaly
› add a new review for Ann with a different email
what’s going on?

<table>
<thead>
<tr>
<th>reviewer</th>
<th>subject</th>
<th>rating</th>
<th>email</th>
<th>ratingstars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chloe Closure</td>
<td>Lucid</td>
<td>3</td>
<td>cc@mit</td>
<td>***</td>
</tr>
<tr>
<td>Chloe Closure</td>
<td>Clover</td>
<td>5</td>
<td>cc@mit</td>
<td>*****</td>
</tr>
<tr>
<td>Ann Alert</td>
<td>Clover</td>
<td>5</td>
<td>aa@mit</td>
<td>*****</td>
</tr>
<tr>
<td>Ben Bitdiddle</td>
<td>Cosi</td>
<td>3</td>
<td>ben@mit</td>
<td>***</td>
</tr>
<tr>
<td>Ben Bitdiddle</td>
<td>Lucid</td>
<td>4</td>
<td>ben@mit</td>
<td>****</td>
</tr>
</tbody>
</table>

spurious redundancy
› don’t need to repeat emails

spurious relationships
› review is not associated with email address except through user
normal forms

criteria for avoiding these anomalies
› by Codd, inventor of relational model

first normal form
› entries are scalars and not sets
› enforced by relational DB language

second normal form
› no field depends on just part of a key
› eg, key for reviews is (reviewer, subject)
   but email depends on reviewer alone

third normal form
› no field depends another field but not on the key
› eg, ratingstars depends on rating alone
database normalization

database schema

in normal form?

yes

done

no

fix schema by adding tables
is normalization useful?

for identifying & fixing problems
  › yes: gives vocabulary & diagnosis

for designing schemas from scratch
  › no: how do you get the first schema?

and should all DBs be normalized?
  › no: redundancy can be useful

read/write ratio
  › suppose many reads and few writes
  › then may be better to avoid joins
a different approach

a hint
▪ normalization takes you to more intuitive schema

so why not start with intuitive structure?
▪ based schema on object model
▪ conceptual model of domain