COMP 430
Intro. to Database Systems
Design Patterns
Design patterns from a practical viewpoint

Most of these ideas previously mentioned in modeling examples.
Special uses of one-to-many
Disjoint groups – Rice students & colleges

Each student belongs to one college.

College(college_id, name, founding_year)
Student(student_id, name, college_id)
Shared disjoint groups – O-Week groups

Each new student belongs to one O-Week group.
Each advisor belongs to one O-Week group.

OWeekGroup(name)
NewStudent(student_id, group_name, room, is_transfer)
Advisor(student_id, group_name, major)
Lookup/Validation table – Rice colleges

A table that lists all the valid data for some attribute. Referential integrity provides guarantee.

Version 1:
- Student(student_id, first_name, last_name, college_name)
- College(college_name)

Version 2:
- Student(student_id, first_name, last_name, college_id)
- College(college_id, name, abbreviation)
Common many-to-many variants
Plain ol’ many-to-many

“Main” tables are unrelated.
Related via junction table with combination primary key.

Course(crn, dept_code, course_number, title)
Student(student_id, first_name, last_name)
Enrollment(crn, student_id, grade)
Many-to-many with (synthetic) key

Allows combination of foreign keys to be repeated.
Simpler primary key – easier to index, easier to relate to another table.

Course(crn, dept_code, course_number, title)
Student(student_id, first_name, last_name)
Enrollment(id, crn, student_id, grade)
Many-to-many-to-many...

Multiple tables can be connected to a junction table.

Course\( (\text{crn, dept\_code, course\_number, title}) \)
Student\( (\text{student\_id, first\_name, last\_name}) \)
Semester\( (\text{year, session}) \)
Enrollment\( (\text{crn, student\_id, semester, grade}) \)

<table>
<thead>
<tr>
<th>session</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall</td>
</tr>
<tr>
<td>Spring</td>
</tr>
<tr>
<td>Summer1</td>
</tr>
<tr>
<td>Summer2</td>
</tr>
<tr>
<td>Summer3</td>
</tr>
</tbody>
</table>
Combination of many-to-many’s

Multiple entities/tables can be related in a more complex way. A junction table can itself be in a many-to-many relationship.

Product(product_id, name, category)
Customer(customer_id, first_name, last_name)
Salesperson(sales_id, first_name, last_name)
Purchase(purchase_id, customer_id, sales_id, date)
PurchaseProduct(purchase_id, product_id, quantity)
A common confusion – disjoint groups vs many-to-many