CS 331
- Kangmei Yang
Relational Model

Translate the conceptual model into logical data model. Validate this model to check it’s structurally correct and able to support the required transactions.

Use Relational Model for Relational Database
Relational Database – a collection of normalized (appropriately structured) relations, all data is in terms of tuples grouped into relations
Relation – table /file – Entity (ER)
Attribute – named column /field
Domain – the set of allowed values for one or more attributes
Tuple – row of record (extension/state of a relation)
Degree of a relation – number of attributes
Cardinality of a relation – number of tuples
Relation schema

- Structure of a relation
- Named relation defined by a set of attribute and domain name pairs
- In relational database schema, each relation has a distinct name
- Each cell of relation contains exact one (atomic) value
- Each attribute has a distinct name
- Values of an attribute from same domain (domain integrity)
- No duplicate tuples
- No ordering of attribute or tuple
Null Value

- Not Applicable
- Not exist
- Not known

Primary Key can’t be null (Entity Integrity)

Foreign Key
- Set of attributes in one relation matches the candidate key of other (possible to be the same) relation, usually represents a relationship

FK must match a candidate key or null – wholly (Referential Integrity)
Translate

Given ER diagram
Translate to Database Schema

*derived relation – view

Identify Relation and write relation schema
RN(kAttri, attri2);

Strong Entity, has its own PK
Weak Entity, PK from owner
1:* or *:1 relationship, FK on many side
1:1 relationship
- Mandatory participation on both sides
  - Combine into 1
- Mandatory participation on one side
  - FK on mandatory side
- Optional participation on both sides
  - Define parent/child, FK on child side
Translate con’t

Specialization (sub class)
- Mandatory, and
  - Single relation
- Mandatory, or
  - One relation for each combine attributes
- Optional, and
  - One relation for superclass and one relation for each subclass
- Optional, or
  - One relation for superclass and one relation for each subclass

*::* – extra entity for relationship, PK from each relation

Complex relationship
- extra entity for relationship, PK from each relation
Translate con’t

Multi-valued attributes – extra entity, PK includes parent PK
Keys

Uniquely identifies each occurrence of an entity type
- **Surrogate Key**
  - a make up attributes that uniquely identifies each occurrence

- **Foreign Key**
  - exist in a relationship, referring to an entity occurrence that it has a relationship with, it’s the primary key of that entity type.

- **Primary Key**
  - a candidate key that is selected to uniquely identify each occurrence, ensures future uniqueness

- **Alternate Key**
Integrity Constraint

- Domain Constraint
  - all values of an attribute must from the attribute’s associated domain
- Entity Integrity Constraint
  - the attribute value that’s part of primary key can’t be null
- Referential Integrity Constraint
  - either the value of any foreign key exits in a tuple in referenced relation, or the entire foreign key value is null.
- Primary Key Constraint
  - No two tuples in a relation have the same primary key value
- Multiplicity
- General Constraints
  - Additional rules from the enterprise
# Data Dictionary

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>studentId</td>
<td>Integer</td>
<td>10</td>
<td>Primary Key</td>
</tr>
<tr>
<td>surname</td>
<td>Text</td>
<td>20</td>
<td>Not Null</td>
</tr>
<tr>
<td>givenName</td>
<td>Text</td>
<td>20</td>
<td>Not Null</td>
</tr>
<tr>
<td>dob</td>
<td>Date</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
Exam Taking

- **Proctor**
  - socSecNum\{PK\}
  - name
  - email

- **ExamTaker**
  - socSecNum\{PK\}
  - name

- **Exam**
  - versionNum\{PK\}

- **Takes**

- **grades**
Exam Taking

Proctor(socSecNum, name, email);
ExamTaker(socSecNum, name);
Exam(versionNum);
TakeExam(exam, student, proctor, grade);
Foreign Key exam references Exam(versionNum)
Foreign Key proctor references Proctor(socSecNum)
Foreign Key student references
ExamTaker(socSecNum)
Department Courses

- Transcript
  - /credit
  - /gpa

- Course
  - num
  - name
  - credit

- Student
  - id
  - name
  - email

- Major
  - num
  - name

- Department
  - code
  - name
Department Courses

Department(code, name);

Major(num, name, department);
Foreign Key department references Department(code)

Student(id, name, email);

MajorDeclaration(student, major);
Foreign Key student references Student(id)
Foreign Key major references Major(num)

Course(department, num, name, credit, preReqC, preReqD);
Foreign Key department references Department(code)
Foreign Key preReqN, preReqD references Course(num, department)
Department Courses

Student(id, name, email);
Derived credit
Derived gpa;

TranscriptCourseLine(student, crs, dept, grade)
Foreign Key student references Student(id)
Foreign Key crs, dept references Course(num, department)
Online Order

- UML

Customer
- customerId {PK}
- name
- address

Order
- orderNum {PK}

Shipments
- quantity

OrderLines
- quantity

Products
- prodId {PK}
- descrp

[0..1]
[1..1]
[0..*]
[1..*]
[0..*]
[0..*]
Online Order

- UML

**Product**
- prodId {PK}
- descrp

**Order**
- orderNum {PK}
- quantity

**Customer**
- customerId {PK}
- name
- address

**Shipment**
- quantity

[1..1] places
[0..*]

[1..*] contains [0..*]
[0..*]

[1..*]
Online Order

The company sells products to customers, who’re identified by CustomerID, also known name and address.

Products are identified by a product ID with description on the product.

Whenever a customer place an order, it generates a unique order number, that it contains the products and quantities that’s been place on the order.

An order may be packed into several shipment which contains the items it ships and the quantity.
Online Order

- UML

Customer
customerId {PK}
name
descrp

Order
orderNum {PK}

Product
prodId {PK}
descrp

OrderLine
prodId {PPK}
quantity

Shipmenqnt
quantity

Order

[1..1]

[0..*]

[0..*]

[1..1]

[1..*]
Online Order
Customer(customerId, name, address);
Product(prodId, descrip);
Order(orderNum);
Shipment(sId, quantity);
OrderLine(order, prodId, quantity, shipping);
Foreign Key order references Order(orderNum)
Foreign Key prodId references Product(prodId)
Foreign Key shipping references Shipment(sId)