Normalization : Denormalization

Objectives of Normalization

- Normalization: techiques presented based on a large body of research into the logical design process
  - Normalized schemas, relations & theme
  - Purpose: to produce relation sets faithful to operations & the enterprise
    - highly flexible
    - extended when needed (addition of entities, attributes, relationships)
    - reduce redundancy
    - avoid inconsistencies (anomaly - inconsistent, incomplete, contradictory state of the DB.

Insertion, Update, and Deletion Anomalies

- Update anomaly: record is updated, but other appearances of the same items are not.

E.f. Codd ID causes and defined the first 3 normal forms

- Insertion Anomaly: user is ~~able~~ unable to insert a new record when it should be able to do so.
- Deletion anomaly: record is deleted, other information that is tied to it is also deleted.

- Normalization: process of transforming a relation into a higher normal form.
  - Clear grasp on semantics of the model.

w/ that are dependencies: functional, multivalued, and join
(FDs) (MVDs) (JD)

Normalizing using Candidate Keys

First Normal Form

(1NF) if and only if, every attribute is a single valued for each tuple. (Atomic)

-Each attribute in each row/"cell" of the table has only 1 value

No sets, array, lists, etc... are allowed in the domain - only single value.

Full Functional Dependency (FD) & Second Normal Form

- In a relation R, attribute A of R is fully functionally dependent on an attribute or set of attributes X of R if A is functionally dependent on X but not functionally dependent on any proper subset of X.

Example:

LastName fd on [classNo, stuId], also FD on proper subset of that combo → StuId

FacId, schedule, room are fully functionally dependent on the combo [classNo, stuId]

2NF: Second Normal Form - if it is in 1NF and all non-prime attributes are fully functionally dependent on all candidate keys

→ non-prime attribute:

an attribute that is not part of any candidate key

ex. Class relation is not 2NF because lastname is not fully functionally dependent on the key [classNo, stuID]

→ Projection transforms a 1NF that is not 2NF into equivalent relations. → performing projections on the original relation is such a way that it is possible to get back the original by taking the join of the projections. (Lossless projections)

1NF to 2NF you ID non full FDs and form projections by removing the attributes that depend on each of the determinants

- Converting to 2NF

- ID each partial functional dependencies
- Remove the partially func dependencies attributes that depend on each of the determinants so ID'd
- Place determinants in seperate tables along w/ their dependent attributes.
- In original table, keep the composite key and any attributes that are fully functionally dependent
- Even if the composite key has no dependent attributes, keep that relation to connect logically the others

ex:

2NF tries to reduce redundant Data getting stored on Disc

- Transitive Dependency; Third Normal Form

Transive Dependency: If A, B and C are attributes of relation R, such that $A \rightarrow B$; $B \rightarrow C$, then C is trasitively dependent on A

(3NF)

→ third normal form, eliminate certain transitive dependencies

- insertion, deletion, and update anomalies.

∴ A relation is in third normal form (3NF) if it is 2NF and whenever a nontrivial functional dependency $X \rightarrow A$ exists, then either X is a superkey or A is a prime attribute.

- Each non-prime att is FD on the entire key, for every CK ; no other att.
- Any non candidate key attr(or group) is FD on another non-key attr or group. - Such FD exist, Remove the FD attr from the relation. placing it in a new relation w/ its determinant.

Boyce-Codd Normal Form (BCNF)

A relation is in BCNF if, a nontrivial FD $X \rightarrow A$ exists, X is the superkey

→ Check BCNF; ID all determinants and verify that they are superkeys. If not, breakup the relation by projection until there is a set of relations all in BCNF.

- Each determinant gets a seperate relation. In BCNF.
- Relation contains all attributes it determines

Ex.

Relation: newstudent

→ determinants stuId, credits

↑ not superkey; the whole relation is not BCNF

Comprehensive Example of Func Dependencies

Ex. Work(projname, projmgr, empid, hours, budget, StartDate, salary, empmgr, empDept, rating.

1.

following FDs

projname → projmgr, budget, startdate

empId → empName, salary, empmgr, empDept

projname, emId → hours, rating

empDept → empmgr

First normal form (1NF)

Composite candidate key. (projname, empId) word → 1NF

Second normal form (2NF) partial dependencies

projname → projmgr, budget, startdate

empId → empName, salary, empmgr, empDept

transform relation into equivelant set of 3NF by projection:

Proj(projname, projmgr, budget, startDate)
Emp(empId, empName, salary, empMgr, empDept)
Work1 (projname, empId, hours, rating)

set of projections
to test 3NF. - proj does not have non-prime attribute/combination of attributes (3NF in proj).

- Emp: transitive dependency empDept → empMgr
  - empDept: not a superkey; not 3NF
  - empMgr
- Work1 has no transitive dependency; in 3NF

↓

Emp1(empId, empName, salary, empDept)
Dept(empdept, empMgr)

↓

proj(projname, projmgr, budget, startdate)
Emp1(empId, empName, salary, empDept)
Dept(empDept, empMgr)
Work1(projname, empId, hours, rating)

Boyce-Codd Normal Form revisited: 3NF of relations is already BCNF, in each relation, the only determinant is primary key, which is also a superkey.

9.2 OO Data Concepts

Obj management group (OMG) (1989)

- obj: corresponds to an entity
  - → has a state & behavior
    - ↓
      - behavior is the set of methods that can be performed on it → internal structure of the obj; descriptive attributes and data elements and the relationships
      - methods used to create, access, and manipulate the object
      - → Signature found in each method, gives its names and the #, order and types of its parameters

Encapsulated objects are: - obj's data & methods form a unit and that access to the data is restricted.

- interface: external appearance of an object, visible to the outside world. → how the outside world interacts w/ the object

Literal:

differs an obj that it has same state (value) w/ no obj identifier (OID)

→ OO systems support user defined datatypes (UDTs)

Class

description or template of obj having same structure → attr w/ data types, same methods & relationships.

Class = operations, relationships

- Data members: attributes, instances, variables
  - → member methods: functions / procedures that belong to the class.

class class-name { list of attributes, relationships, methods }

- Constructor: method that creates new obj of the class; same name as the class.
- mutator method: inserts / updates the value of a data member
  - Calling the obj; calls the method, becomes implicit ~~passed~~ parameter.

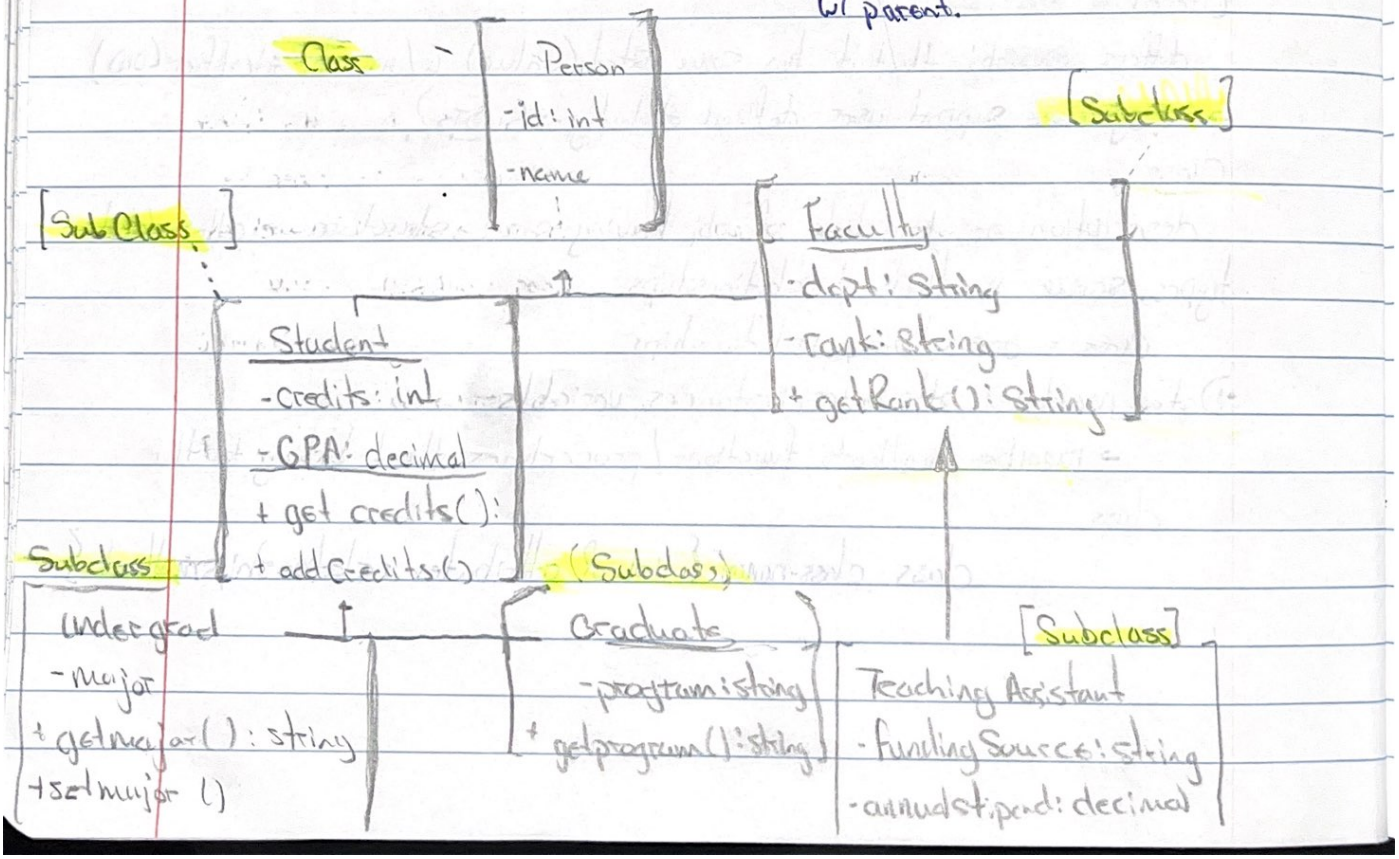
observer method
- returns value of a data member

Overloaded: (method) same name may be used for different classes w/ their perspective code.

Extent - set of obj belonging to a class.

Hierarchies & Inheritance:

- Class hierarchies: tree-like structures
  - Base class: any # of child classes
    - → Parent: Superclass
    - → Child: ~~Subclass~~ Subclass → has an extends relationship w/ parent.



* Override:
- Multiple inheritance: subclass belongs to more than one superclass
  - inherits attributes & methods from the multiple superclasses.

* Obj Identity
- each obj in the DB gets a unique obj ID (OID)
  - unchanged for the lifetime.
  - OO gives unique IDs automatically

* OO Data Modeling Using Unified Modeling Language (UML)
- ER diagrams do a poor job @ representing obj.
  → they do not allow rep of methods

OMG (obj management group) - defines UML → unified modeling language developed as a standard for software models

UML class allows for OO DB concepts to be expressed more naturally.

In UML: 

class
↓
top: class name
↓
middle: attributes
↓
bottom: methods

String in guillements (<<>>) is the stereotype
<<persistent>> class is persistent

- Generalization Hierarchies: repped by lines connecting class to subclass w/ a triangle pointed toward the superclass
- Overlapping subclass ▲ filled triangle - an obj belong to both subclasses, △ outline means disjoint ones.

UML - types of relationships → association: uni/bi-directional relationship between classes..
- directed lines connect rectangles

- Aggregation: connects its whole to its parts "it is part of"

Multiplicity:
→ show cardinality; participation

0...1	zero to one
1	One only
0...*	Zero +
1...*	One +
2..4	Specified range
2, 4...6, 8	multiple, disjoint ranges

- Role Names
appear on association line if needed; appear @ the head of the arrow i.e $\xrightarrow{\text{name}}$

Reflective Association / aggregations
- association line or aggregation → relationship that connects courses and their prereqs is a reflective relationship.
- Weak entities: represented by a box connected to the owner w/
  - discriminator: appearing in a box beneath the owner rectangle

Extending the relational model

- designed to represent data that consits of single valued attributes & a small collection of data types.
  - lack features to represent complex types & relationships needed for advance apps

- Large Object Data types

LOB (large obj)
- restricted func, substring ops
- stored & rarely retrived
- outside
  - → Lob locator: system generated binary surrogated

ex: picture CLOB REF IS picId SYSTEM GENERATED

- Standard Types

Array & Row, set, multiset (bag): Oracle has two collection types:
Varray & Nested table → preform similar functions of SQL
- Associative Array: key-valued pairs

ex:

CREATE TYPE arraytype-name AS VARRAY(size) OF base-type;

- UDTs (User-Defined Data Types

Obj Types: "classes" in OO model; any SQL type
- Column Obj
  - → use a user-defined type for a column of any table.
- Obj Tables
  - → each row = an object
    - ↓
    - row object (has: unique OID

Not Instantiable: no instances of that type can be made; can be used as base types; subtypes "inherit" & subtypes = instantiable
- Not Final: we can create/define subtypes.

Obj View

→ defining obj type in which each attribute corresponds to a column of existing relational table.