Database Ch. 6 notes

* Transforming Data Models into Db Designs *

I. Purpose of a Db Design

Db design: set of db specs that can actually be implemented as a db (DBMS-specific)

II. Create a Table for Each Entity

~~attr. = co~~

- entity attr = table column
- identifier = table PK

Selecting a PK

- facilitates searching and sorting of table rows

* Figure 6-1 * - Transforming Db Model into Db Design
  - pg 250

- ideal PK is short, fixed, and numeric (id)
  - use surrogate key if no candidate keys
- disadv of surrogate keys:
  - a) value has no meaning to the user
  - b) when sharing data, if there's 2 db's that have 2 tables w/ same name, they'll have the same ID rows (not likely) (unless merging db's)
    - surrogate keys can have a defined range of values

Specifying Alternate Keys

candidate key: identifier of unique rows w/in a table

- we choose 1 in a table to be the PK
  - the ones not chosen are alternate keys

- representing AK's: 

AK 1.1 → 1st column of the 1st alternate key

Specifying Column Properties

4 types: ~~null, not null~~, null status, data type, default value, data constraints

null status
- column can either be null or not null

data type
- DBMS - specific
- some are standard across the board
  - Char for fixed-length string
  - Varchar(x) for variable-length string
- companies usually have their own generic data standards

☆ pg 254-258 = Common Data Types in MySQL ☆

Default Value
- a value supplied by the DBMS when a new row is made
  - could be string, date/time, timestamp, int, etc.
  - could be the result of a function (ex. calculating price)

Example: table: ITEM
column: ItemPrefix
Default Value: if Category = 'Perishable' then 'P'
otherwise 'N'

☆ Figure 6-8 pg 259

Data Constraints
- limits on data values (Ex. only 4 values allowed in this col.)
- Range constraints limit values to a particular interval of values
  - Ex. dates b/w 1/1/1900 & present

- Intrarelation constraint limits column values in comparison w/ other columns in same table
    - Ex. ReviewDate must be at least 3 months after HireDate
- Interrelation constraint limits a column's values in comparison w/ other columns in other tables
    - Ex. DEAD. Name column value cannot equal ALIVE. Name column value
  - Referential Integrity is one type of interrelation constraint

Verify Normalization

- structures of forms/reports usually reflect how users think about their data
    - will usually end up normalized if modeling from forms/reports
- should verify though

III. Create Relationships

*Prerequisite: design a complete, but independent, set of tables

Relationships b/w Strong Entities

- generally, relationships are created by placing foreign keys into tables
- 1:1, 1:N, N:M

1:1 Relationships

- ~~Option 1~~: place PK of Table1 as FK of Table2, or vise versa
    - each value in PK column of T1 can only appear once in T2

- To enforce required uniqueness of the FK value, we can define the FK column as 'unique', either directly or as an alternate key
    - this is also how we enforce maximum cardinality

1:N Relationship

- place PK of T1 on the 1 side (1:N) into the Table on the many side (1:N) as a FK
- 1:N
  parent, child
    - Don't make the FK unique
- "Place the PK of the parent as the FK of the child"

* Figure 6-10 pg 261

N:M Relationships

- problem: no place in either table to place the FK
    - a 2-way 1:N rel won't work (no duplicates!)
- solution: create an intersection table, holding the PK's of each table only, as FK's
    - *Figure 6-11 pg 262
- the 2 PK's in the intersection table serve as the composite PK
    - ints. table holds key data only

Relationships using ID-Dependent Entities

4 purposes:

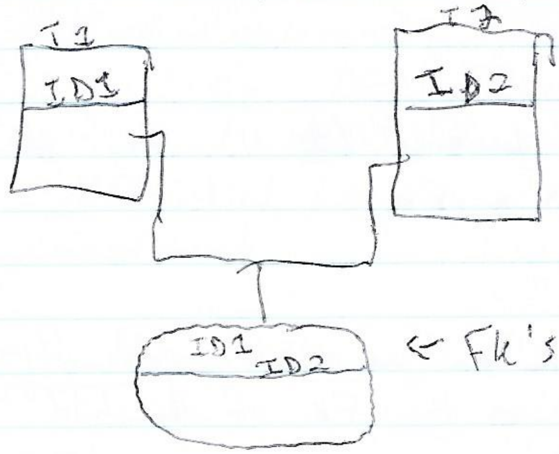
1.) Repping N:M relationships (intersection table = ID-Dep)
2.) Repping association relationships
3.) Storing multivalued attributes
4.) Repping archetype/instance relationships

CN | AN | PN | HW
1 | null | null | 40

*ask about Figure 6-14

Association Relationships

Start: - intersection table that's ID-Dep on both its parents
- convert to association table by adding non-identifier attributes



- can connect >2 tables in intersection table

Multivalued Attributes

See Figure 15-a and 15-b (pg 267)

Archetype / Instance Pattern

- if the pattern is given identifiers of their own, it becomes weak but no longer ID-Dependent
  - relationship must then be transformed using rules of a 1:N rel. b/w a strong & weak/non ID-Dep entity
  - ↑ means the PK of parent table should be placed in the child table as a FK

Relationships w/ a Weak, Non-ID Dependent Entity

- relationship b/w 2 strong entities behaves the same
- nonidentifying rel & is characterized by max card.
- use surrogate key to replace long identifier situations (paren
  - then replace identifier of ID-Dep entity w/ it's own surrogate key

Relationships in Mixed Entity Designs

Mixed-ID: strong entity + ID-Dependent entity
*Figure 18 + Figure 19 (pg 269) show great examples of how these relationships work

Relationships b/w Supertype + Subtype Entities

- 'IS-A' Relationships bc the supertype + subtype represent the same underlying entity
    - Eg a MANAGER is an EMPLOYEE
    - keys of all subtype tables are same as keys of the supertype

- discrimator attr cannot be repped in relational designs
    - *Figure 6-20 (pg 270)

Recursive Relationships

1:1

- Boxcar data model = Figure 6-21 (pg 271)
- create a FK in child table that contains an identifier of its recursive child
    - FK must be unique
- both sides of the relationship are optional (no required children) bc of the first + last boxcar

1:N

- All cases: Place the PK of parent table in child table as a FK
    - if ~~the~~ a data model requires children (i.e. circular), then each parent table has a child + vise versa

N:M

- *trick*: decompose N:M into 2 1:N relationships (i.e. create intersection table)

Representing Ternary & Higher-Order Relationships

- ~~ea~~ ternaries could be repped by multiple binary rels (2 tables)
    - issue if a business rule is added that affects the relationship (i.e. particulars)
    - the ability to only allow certain combos in a table can only be enforced by program code
- MUST constraint: requires one entity to be combined w/ another entity
- MUST NOT constraint: combos that aren't allowed in a ternary rel
- MUST COVER constraint: all combos that must appear in a ternary rel

- none of these 3 types of binary constraints can be represented in the relational design

Relational Representation of Highline Univ. Data Model

- Final data model: *Figure 6-27 (pg 275)
- Final db design: *Figure 6-28 (pg 276)
    - look at the design & visualize the tables... it makes sense!
    - note: cannot have 2 columns (PK's & FK's) w/ same name from 2 different tables

IV. Design for Minimum Cardinality

- complicated bc certain things cant be enforced in db structures
- Rels can have 4 min. cardinalities: parent/child optional (O-O); parent mandatory & child optional (M-O); O-M; M-M

- (0-0) rels dont require enforcement of min. card.
  - *Figure 6-29 : actions needed to enforce minimum cardinality
    - (a): When parent is Mandatory
    - (b): When child is mandatory
  - "minimum cardinality enforcement action"
  - actions are for Insert, Update, + Delete functions

Actions when Parent is Required
- ensure every row has a non-null value for FK column
  - restrict action to update or delete parents PK and restrict creating or modifying child FK

Actions on Parent row when Parent is required
- if update:
  - if any value of parent PK changes, then the child FK becomes an orphan (prohibit modification)
    - be consistent in your changes

Cascading Updates: policy of making changes happen from Parent PK to child FK
- if delete:
  - delete PK row = children FK are orphans
    - either delete children too or prohibit deletion

Cascading Deletions: deleting the children along w/ the parent

Actions on Child row when Parent is required
- child must have valid FK value
  - could have a default value for the FK
- no restrictions of deleting a child

Actions when Child is Required
- ensure there's at least 1 child row at all times
  - last child cannot leave parent

- enforcing required children is harder than enforcing required parents
    - we must count # children a parent has

Actions on Parent Row when Child is required

- if child is required, then can't create a new parent w/o creating a relationship to a child
    - either create new child row each time parent is created or change an existing child row's FK
        - else = prohibited
    - modification of parent's PK has no restriction if parents use surrogate key
    - if child is required & parent is deleted, no action need be taken bc. child is required, not the parent

Action on Child row when Child is required

- no action w/ insertion
- if child is the last of its parent, then update cannot occur; if so, parent would be childless
    - must know # of children at all times
- if last child, cannot delete child row

Implementing Actions for M-O Relationships

- never create orphans!

a) - enforce referential integrity so every FK value has a match in a PK table

b) - make FK column NOT NULL
    - if a + b, then all of *Figure 6-29 will be enforced

*Figure 6-31 (pg 280) - Actions to Apply to enforce Minimum Cardinality

Implementing Actions for O-M Relationships

triggers: modules of code that're envoked by the DBMS when specific events occur
- used for insert, update, delete actions
- use cases:
  - parents - create the required parent or child
  - steal existing children from parents
  - children - on updating / deleting parents, if FK is null, proceed. If FK has a value, and the row is the ~~last~~ last child, the trigger must either:
    - Delete the parent
    - Find a substitute child
    - Disallow (reject) an update / delete
- triggers require code logic

Implementing Actions for M-M Relationships
- hard to enforce
- Ex: 2 tables; each has an insert trigger; it calls insert on T1, but fails bc trigger requires a row in T2 before an entry in T1 can be made, and vise versa.
  - similar issue in deleting

Designing Special-Case M-M Relationships

- (M-M's) are easier b/w strong & weak entities
  - "no one will ever try to insert, update, or delete a new child except in the context a the parent
- all actions apply for Action on Parent from Figure 6-29 (a & b)
  - insert = always create child
  - update / delete = cascade remaining actions

Documenting Minimum Cardinality Design

Doc. Required Parents

- 3 design decisions to be made:
    1.) determining if an update to parent's PK should cascade or be prohibited
    2.) same as 1, but w/ deletion
    3.) identifying how a parent row is to be selected on insertion of a child

Doc. Required Children

- use Figure 6-29 (b) as a boilerplate document (i.e. copy the figure for each rel. w/ a required child to fill in specific actions for C~~R~~UD.

An Additional Complication

- need to specify a design for the minimum card. of every relationship (it'll vary b/w O-M, M-O, & M-M)
- triggers; some tables will have > 1 for each C~~R~~UD function

*Figure 6-34 (pg284) - Summary of design desisions for min. cardinality

V. View Ridge Gallery Db

- Example design (starts pg. 284
- the gallery sells a ton of stuff
    - Figure 6-36 (pg 286) has the requirements list
    - Figure 6-37 (pg287) shows the data model
    - Figure 6-38 (pg 287) shows the db design & Figure 6-39 shows the final design

Figure 6-40 (pg 288) shows all relationships:

P	C	Type	Max	Min
Artist	Work	nonidentifying	1:N	M-O
Work	Trans	" "		M-M
Customer	Trans	" "		O-O
Customer	Cust_Art	identifying		M-O
Artist	Cust_Art	" "	" "	M-O

- Figure 6-41 (pg 289) shows Actions to enforce Minimum Cardinality for required parents
- Figure 6-42 (pg 290) shows " " for Work-To-Trans relationship
* - Figure 6-43 (pg 291) shows all column properties

Database Ch. 8 notes: Database Redesign