- 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

- the 3rd reason db's can be created is db redesign

I. The Need For Redesign

a) Could've been designed incorrectly the 1st time
   - especially common in large enterprise dbs
b.) as user behavior needs to change with the changes made in the information system of the organization, the db may ~~need~~ need to be updated in it's design

- this is the system maintenance step of SDLC in the systems analysis & design process (of info systems)
   - changes & redesign are going to happen no matter what; we must be prepared

II. SQL Statements for Checking Functional Dependencies

- redesign is hard w/ lots of already-existing data
- find all issues B before making 1 change

2 SQL helpers:
1.) correlated subqueries (i.e. nested query loop)
2) SQL EXISTS & NOT EXISTS ~~stmts~~ keywords

1a.) correlated subqueries are when the bottom, cannot be ran before the top select statement; rather they're ran concurrently
select stmt

- Kind of works like a nested loop

Example:

```
Select  W1.Title, W1.Copy
From    WORK as W1
Where   W1.Title IN
        (Select  W2.Title
         From    WORK as W2
         Where   W1.Title = W2.Title
         And     W1.WorkID <> W2.WorkID);
```

call same column here

like a new instance of the WORK table

Logic: - take row 1 from W1
- evaluate the subquery (W2). To do this we compare each row in W2 w/ the 1 title from W1. Once all rows from W2 have been ran against the first row from W1, grab W1's next row

Logic for next query loop

Using Corr. Subq. to check for Funct. Dep.
- run similar query as above
pg: 400 - 401

Correlated Subq. w/ EXISTS & NOT EXISTS

- these operators test whether there are any values returned by the subquery, indicating conditions are being met
  - values returned from the subquery are used to run against the top query

- Works again like a nested loop; ~~if any EXIST returns true, it's kept and used against the top query~~

NOT EXISTS in a Double Negative

- NOT EXISTS will only be true if all rows in subquery fail the conditional
  - can be used to find rows that do not, not match a condition

Ex. pg402 - if a row doesn't match any row, it matches every row
- Suprisingly used often ??

III. How Do I Analyze an Existing Db?

Example: "I want you to change the PK of Table 1."
checks:
- how many foreign keys were used w/ the PK
- do any Views use the PK?
- do any triggers / stored procs use the PK?

This could turn into a maaajor project
- be very gentle (make a list & check it twice)
- get a test db (sandbox) ready to test the actions in
- create a backup db (prod) if possible as insurance

Reverse Engineering

- process of reading db schema & making a data model from it
  - it's a table-relationship diagram dressed in entity-relationship clothes (Figure 8-3 pg404)
- Logical = data model
- Physical = db design

- MySQL Workbench only makes db designs?

Dependency Graphs

- diagrams w/ nodes connected to other nodes by lines
- Figure 8-4 (pg405)
  - shows all dependencies (references) to a specific entity, attr, etc.

Db Backup & Test Db's

- do this prior to any changes being made
- thorough testing procedures (3 copies of db schema for testing)

- test envr. must be able to revert easily
- smaller test db's must still have same characteristics of operational db
- for large organizations, it may not be possible to make a complete copy of the prod db.
- very large job todo! There's jobs for it out today just for making test db's

IV. Changing Table Names & Table Columns

Table Names

- there will be a lot of references to that specific name throughout the db and its functions
- MySQL has a Stmt: RENAME {Name1} to {Name2}
  - this doesn't modify triggers or stored procs
- nuclear solution:
  1.) new table w/ all same structures
  2.) drop old table once all is working in the new

Example: pgs 406-407

Adding / Dropping columns

- if you add a DEFAULT value after data is already in the table, all prior column values will still be null

Adding NOT NULL columns

- SQL stmt will fail if the column hasn't been given values in all rows
- to drop a Fk column, 1st the constraint defining the FK must be dropped (same as PK) → also, all FKs that use the PK = droppe

Changing a Column Data Type or Column Characteristics

- may cause data loss
- converting likewise types (i.e. Char to Varchar) is usually fine

V. Changing Relationship Cardinalities (common)

Changing Minimum Cardinalities

If Parent entity:
- matter of ~~changif~~ changing whether null values are allowed for the FK repping the relationship
- depending on the DBMS, the FK constraint can be dropped & readded before and after the change to the parent
  Ex. Moving col's Min. card. from zero to one would mean changing the col from null to not null

If Child entity:
- must write triggers ~~(appl. code)~~ to change from 0 -> 1
- if going from 1 -> 0, just remove the trigger enforcing the constraint
- dont use appl. code to change/enforce constraints

Changing Maximum Cardinality
- preserve existing relationships

From 1:1 to 1:N
- decide parent entity (since 1:1, parent doesn't matter)
- drop the unique constraint from child entity

From 1:N to N:M
- create intersection table w/ correct FK constraints
- drop old FK column

Reducing Cardinalities (w/ Data Loss)
- to reduce from N:M to 1:N, do opposite of
- to reduce from 1:N to 1:1, ~~do opposite of~~
  - ~~drop~~ make FK values unique, then define a unique constraint on it
- many changes will need to be made to much data could potentially be lost.

VI. Adding / Deleting Tables & Relationships

- creating a table is like normal; just dont forget Fk and Fk constraints
- deleting tables means dropping Fk constraints, then dropping the tables
- construct Dependency Graphs once removed to understand which programs will be affected by the deletion

• this is all for the purpose of either normalization or denormalization

VII. Forward Engineering

- many skeptics dont use redesign tools bc its automated & the importance level to make the correct changes is very high
- automation may not be the best idea when dealing w/ lots of data (write my own SQL)