Relational Model: Current Summary

- Data driven not design driven
  - designed once; data changes over time without affecting applications
  - rules/constraints control how data defined and enforced
- SQL query language
  - Define schema
  - Query data
  - Constraints, triggers, assertions
- Next: how to design schema
  - changes to database scheme without affecting application?

Example

- Database to store information about Customers
  - One table
  - Cust(Last-name, First-name, SSN, address, telephone number)
- Suppose we want to add other types of telephones - such as mobile etc.??
  - Change schema?
A better design – exploiting relational model

- Cust(Last-name, First-name, SSN, address, telephone number)

- Three table schema:
  - Cust (Last-name, First-name, SSN, Address)
  - Cust_Phone(SSN, number, phone_type)
  - Type_Detail(Type, Description)
    - example: (o, office), (h, house), (c, cell)

Relational Model: Definitions Review

- Relations/tables, Attributes/Columns, Tuples/rows
  - Attribute domains
- Superkey
- Key
  - No two tuples can have the same value in the key attribute
  - Primary key, candidate keys
  - No primary key value can be null
- Referential integrity constraints
  - Foreign key
Relational Schema Design

- Logical Level
  - Whether schema has intuitive appeal for users
- Manipulation level
  - Whether it makes sense from an efficiency or correctness point of view

Functional Dependencies and Normal Forms

- Guidelines for database schema design: how to design a “good” schema?
- Example of a COMPANY database: two possible designs to represent Employees and Department information
  S1: EMPLOYEE(LNAME, FNAME, SSN, DNO)
      DEPT(DNUM, DNAME, MGRSSN)
  S2: EMPDEPT(LNAME, FNAME, SSN, DNUM, DNAME, MGRSSN)

Which one is better? S1 or S2?
Functional Dependencies and Normal Forms

- Informal methods
  - Rules of thumb, intuitive reasoning, experience
- Formal methods
  - Provable properties
  - Involve concept of Functional Dependencies
  - Develop theoretical model to define what we mean by “good schema”

Informal Guidelines: 1

- 1: Try to make user interpretation easy
  S1: EMP(FNAME, LNAME, SSN)
      WORKS_ON (SSN, PNO)
      PROJECT_LOC(PNO, PLOC)

  S2: EMP(FNAME, LNAME, SSN, PNO, PLOC)

- Perhaps S2 has too much information to absorb per tuple?
Informal Guidelines: 2

- Try to reduce redundancy
  - In S2 in previous example, suppose only few projects
    - PLOC is unnecessarily repeated too often
  - On the other hand, S1 repeats SSN in WORKS_ON
    - But SSN is a smaller attribute than PLOC (which may be a large string)

Informal Guidelines: 3

- Try to avoid update anomalies
  - Avoid having to search through entire table during update operation
    - Insert, delete, update/modify
  - Avoid losing information
**Example**

S1: EMPLOYEE (ENAME, SSN, BDATE, ADDR, DNO)
DEPT (DNUM, MGRSSN, DNAME)
WORKS_ON (SSN, PNO, HOURS)
PROJECT (PNUM, PLOC)

S2: EMP_DEPT(ENAME,SSN,BDATE,ADDR,DNO,DNAME,MGRSSN)
EMP_PROJ (SSN,PNUM,HOURS,ENAME,PNAME,PLOC)

Both schemas have same attributes.... Problems with S2 ??

---

**Insertion Anomalies in S2**

- Consider inserting information “John Smith works in Department 5”
  - i.e., < John Smith, 123456789,…,5,Research,111223333>
  - What do we need to check in the table to maintain correctness ?
- Check DNAME is correct – i.e., all tuples with DNO=5 have Research and 111223333
  - Scan the whole relation since DNO is not a key!
Insertion problems…contd.

- Consider creating a new department in the Company: DNO=9, DNAME = ‘Sales’
- Only one way to do this: create NULL values for employee info
  - But that means NULL value in primary key (SSN)!!

Deletion and Modification Anomalies

- What if we delete the last employee in ‘Research’ department
  - Eg. (John Smith, 123456789,…,5,’Research’,…)
- We lose the information that Department 5 is Research department
  - Problem: We can then insert a tuple with DNO=5 and DNAME= Sales!!!
- Similar cases for Modification
  - Change Manager SSN of department 5 = change for all Department 5 employees …scan of entire table
Informal Guidelines

- Avoid too many NULL values
  - Space is wasted
  - Problems occur when using aggregate functions like count or sum
  - NULLs can have different intentions
    - Attribute does not apply
    - Value unknown and will remain unknown
    - Value unknown at present

Informal Guidelines: Spurious Tuples

- Split a table into smaller tables (with fewer columns in each) – sometimes a better design in our examples
  - How to split?
- When reconstructing the “original” data, should not introduce spurious tuples
### Example: Spurious Tuples

<table>
<thead>
<tr>
<th>S1: CAR (ID, Make, Color)</th>
<th>123</th>
<th>Toyota</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2: CAR1 (ID, Color)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAR2 (Color, Make)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What happens when we join CAR1 and CAR2?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>123</th>
<th>Toyota</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>456</td>
<td>Audi</td>
<td>Blue</td>
</tr>
<tr>
<td>789</td>
<td>Toyota</td>
<td>Red</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>123</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>456</td>
<td>Blue</td>
</tr>
<tr>
<td>789</td>
<td>Red</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>123</th>
<th>Blue</th>
<th>Toyota</th>
</tr>
</thead>
<tbody>
<tr>
<td>456</td>
<td>Blue</td>
<td>Audi</td>
</tr>
<tr>
<td>789</td>
<td>Red</td>
<td>Toyota</td>
</tr>
</tbody>
</table>

CS 2441
Summary of Problems

- Insertion, Deletion, modification anomalies
- Too many NULLs
- Spurious tuples – called non-additive join
- We need a theory of schema design
  - Functional dependencies and normalization
- Using functional dependencies define "normal forms" of schema
  - A schema in a “Third Normal Form” will avoid certain anomalies