Relational Database Design: Basic Normalization and implementation in MySQL

140.636

F. J. Pineda

Mechanically mapping each entity in an ER diagram into a table in a relational database

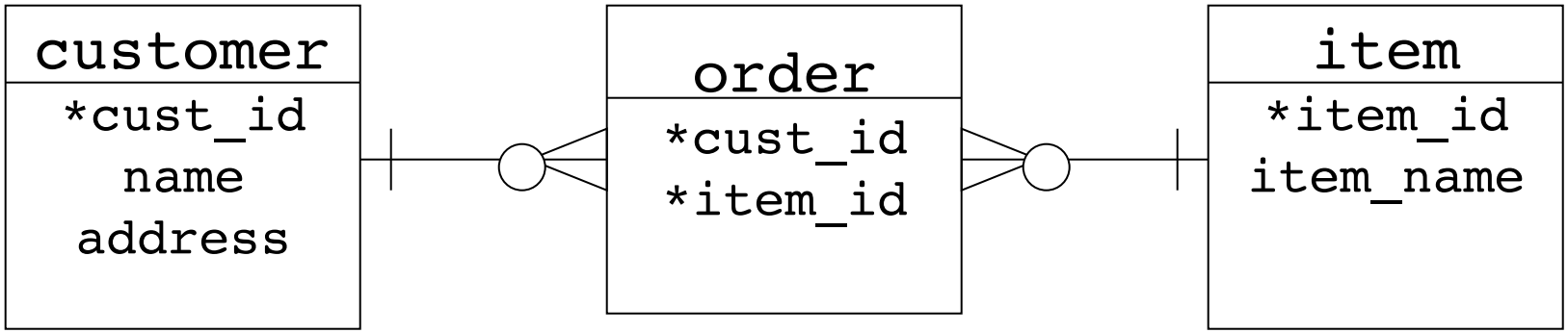
1. Create one table for each entity

2. For each entity that is not at the "many" end of any relationships, create a single-column primary key and use an arbitrary sequence number or appropriate "natural key".

```
create table example (
        example_id int not null auto_increment primary key,
        etc…
);
```

Mechanically mapping each entity in an ER diagram into a table in a relational database

3. For each entity that is at the "many" end of one or more relationships
   - include the primary keys of each parent entity, as foreign keys.
   - If the entity has a natural primary key, use this single column as the primary key, otherwise concatenate as many foreign keys and columns as needed to create a primary key.

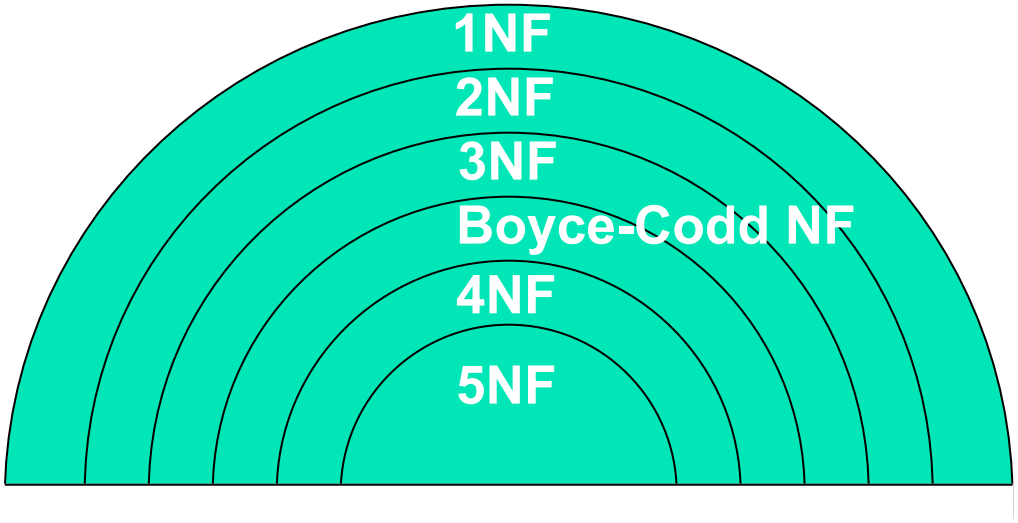



```
create table order (
        cust_id int not null references customer(cust_id),
        item_id int not null references customer(item_id),
        primary key(cust_id, item_id)
);
```

Normalization

- We use ER diagrams to represent database schema, but a schema needs to satisfy some design principles, before it can be considered to be a good database design.
- **Normalization** is a formal processes of eliminating bad design elements from database schema.
- **Normal forms** are a hierarchy of constraints that a schema must satisfy in order to avoid certain well-characterized design flaws (anomalies).
- More often than not, good database design is an iterative and (unfortunately) never-ending process.

Hierarchy of Normal forms



- The normal forms are a hierarchy of rules. If your database satisfies a given normal form, it automatically satisfies all the “less constrained” normal forms.
- 1NF-3NF are a minimum set of normal forms to strive for.

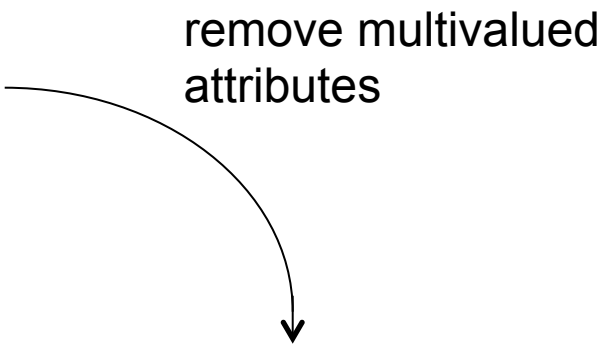
1st Normal form (1NF)

- A database is in **1NF** if-and-only-if there are no multi-valued attributes and no repeating groups
- how to fix this?

ID	Parent	children
1	Fernando	Katy, Daniel
2	Rosie	Lucky,Two,Yeller, Little-Squirt,Furzy

How not to remove multiple valued attributes

ID	Parent	children
1	Fernando	Katy, Daniel
2	Rosie	Lucky,Two,Yeller, Little-Squirt,Furzy



ID	Parent	child1	child2	child3	child4	child5
1	Fernando	Katy	Daniel	NULL	NULL	NULL
2	Rosie	Lucky	Two	Yeller	Little-Squirt	Furzy

A table in 1NF

- The right way to fix multiple valued attributes: add rows

ID	Parent	child
1	Fernando	Katy
1	Fernando	Daniel
2	Rosie	Lucky
2	Rosie	Two
2	Rosie	Yeller
2	Rosie	Little-squirt
2	Rosie	Furzy

Problems with a table in 1NF

- Consider the following table. The unique key of the table is (**custID**,**itemID**)

Orders

cust_id	customer_name	address	item_id	item_name
1	John Smith	615 N. Wolfe	19	pencils
1	John Smith	1899 W. 32nd St.	1074	paper
2	Mike Beach	6 Walker Blvd.	21001	cup
3	Jerry Miller	1901 Rush St.	892	chair
3	Jerry Miller	1901 Rush St.	1074	paper
3	Jerry Miller	1901 Rush St.	97	desk
7	Joe Blow	110 E. 20th St.	98	desk

Problems with 1NF Anomalies

- Insertion anomalies
  - Can't add data about customer until customer orders an item, because otherwise itemID would be null (so have incomplete key)
  - Can't add data about about items for the same reason.
- Deletion anomalies
  - If we delete **`cups`** from the items list, we delete information about the customer **`Mike Beach`**.
  - If we delete **`Johns Smith`** we lose the fact that we have **`pencils`** in our inventory
- The basic problem is that we have multiple entities in a single table.
- We want 1-entity 1-table. We do this systematically by considering *functional dependency* of attributes

2nd Normal Form (2NF)

- Define: Functional dependence
  - An attribute is **functionally dependent** on another attribute (A -> B) if, for each value of an attribute A, there is exactly one value of attribute B.
  - This is the usual property of a mathematical function. The attribute that is analogous to the "independent variable" of a function is called the "determinant."
  - Example:
    - In the **`orders`** table, **`address`** and **`customer_name`** are functionally depend on **`customer_id`**, but **`item_name`** is not.
- A table is in **2nd normal** form if-and-only-if
  - It is in 1st normal form
  - All non-key attributes are **functionally dependent** on the *entire* primary key
- To fix these anomalies we break up offending tables into tables that are in 2NF
- Choosing entities with only functionally dependent (non-key) attributes is as much art as science since it requires intimate knowledge of the problem domain.

Fixing the order table so it satisfis 2NF

Customer

custID	customer_name	address
1	John Smith	615 N. Wolfe
2	Mike Beach	6 Walker Blvd.
3	Jerry Miller	1901 Rush St.
7	Joe Blow	110 E. 20th St.

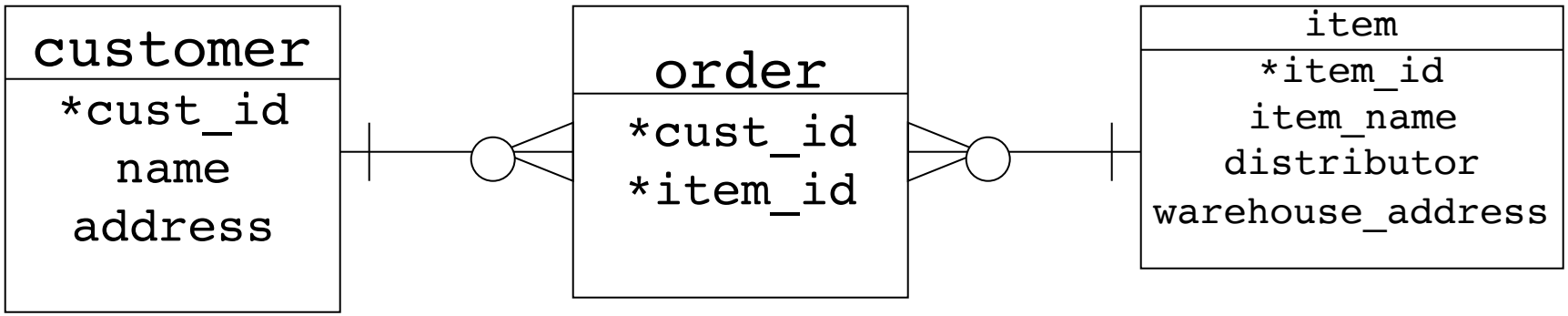
Item

itemID	item_name
19	pencils
1074	paper
21001	cup
892	chair
97	desk

order

CustomerID	itemID
1	19
1	1074
2	21001
3	892
3	1074

2NF states how the non-key attributes depend on their keys



- The nonkey attributes in **`customer`** `table` (`name` and `address`) and nonkey attributes in **`item`** table (`item_name`) are functionally dependent on their respective keys
- Primary key for **`order`** is concatenation of **`cust_id`** and **`vid_id`**

Problems with 2NF

- Consider the following table

item
*item_id item_name distributor warehouse_address

- All non-key attributes depend on the key, therefore table is in 2NF
- But insertion anomaly still exists: Cannot insert data about a distributor until an item exists, or insert NULL item
- But deletion anomaly still exists: If a distributor only has one item and you delete that item, then you delete the distributor as well.
- The problem is that Transitive dependencies can cause anomalies

Third Normal Form (3NF)

- In the previous example, the warehouse address depends on the item

  item_id -> warehouse_address

- but it only depends on the item through the distributor.

  item_id -> distributor-> warehouse_address

- therefore we have a **transitive dependence**
- 3NF has no transitive dependency
- We break up offending tables into two tables

item
*item_id
item_name
dist_id

item_id -> item_name
item_id -> distr_id

distributor
*dist_id
distributor
warehouse_address

dist_id -> distributor
dist_id -> warehouse_address

Higher Normal Forms

- Higher normal forms
  - BCNF (Boyce-Codd Normal Form)
  - 4NF
  - 5NF
- They handle situations that do not commonly occur
- We will not worry about these…
  - In practice if you get to 3NF you are in pretty good shape, if your database is more complex than this, it might be time to hire a professional database developer!

The bottom line

- Put your database in 3NF
- There is some overhead associated with performing joins so normalized databases generally are slower than unnormalized databases
- Only if performance becomes an issue, consider optimizing by creating tables that are fast to query, but are not in a normal form

Database design tools

- Visual database design systems integrate database design, modeling, creation and maintenance
- Specific Tools
  - Designer 10g
    http://www.oracle.com/technology/products/designer/index.html
  - Rational Rose
    http://www-01.ibm.com/software/rational
  - ERwin Data modeler
    http://www.ca.com/us/products/product.aspx?id=260
  - DataArchitect
    http://www.thekompany.com/products/dataarchitect
  - DBDesigner 4
    http://www.fabforce.net/dbdesigner4/
  - MySQL Workbench 5
    http://dev.mysql.com/downloads/workbench/5.0.html

DBDesigner 4/ MySQL Workbench 6

- DBDesigner 4
  - License: GNU GPL
  - Supported Platforms: Windows/Linux
- MySQL Workbench 6
  - License: GNU GPL
  - Supported Platforms Windows/Linux/OSX