Introduction to the Relational Data Model

Beginning Database Design: From Novice to Professional, Second Edition

by Clare Churcher Published by Apress, 2012

Relational Database Design and Implementation, 4th Edition

by Jan L. Harrington Published by Morgan Kaufmann, 2016

The Relational Data Model

We will translate our ER diagram (or logical schema) into a **formal relational data model,** which can then be implemented on a DBMS.

"A **relational database** is a database whose logical structure is made up of nothing but a collection of **relations."** Harrington

A **relation** is like a **table,** with **columns** and **rows,** that has special properties.

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

The Relational Data Model

Example of a relation:

Customer Number	First Name	Last Name	Phone
0001	Jane	Doe	(555) 555-1111
0002	John	Doe	(555) 555-2222
0003	Jane	Smith	(555) 555-3333
0004	John	Smith	(555) 555-4444

Each column is an **attribute** of the relation.
Each row is an **instance** of the relation.

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Relation (or Table) Properties

- Column names must be unique within a relation.
- Columns must have a domain, and values within a column must all belong to that domain. Examples: string, integer, date, boolean, etc.
- The order or position of columns does not matter.
- The order or position of rows doesn't matter.
- Multi-valued attributes are **not allowed**.
- Each relation has a **Primary Key**, which is a **column or combination of columns** with values that uniquely identify each row. ***No two rows may have the same value for primary key.***

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Relation Notation

To represent a relation, we will use a shorthand that looks something like this:

customer (customer_number, first_name, last_name, phone)

A column can be referenced

like this: **customer.first_name**

Customer Number	First Name	Last Name	Phone
0001	Jane	Doe	(555) 555-1111
0002	John	Doe	(555) 555-2222
0003	Jane	Smith	(555) 555-3333
0004	John	Smith	(555) 555-4444

Relation Notation

To represent a relation, we will use a shorthand that looks something like this:

customer (customer_number, first_name, last_name, phone)

Primary Key column that contains a unique value for each row is underlined

A column can be referenced

like this: **customer.first_name**

Customer Number	First Name	Last Name	Phone
0001	Jane	Doe	(555) 555-1111
0002	John	Doe	(555) 555-2222
0003	Jane	Smith	(555) 555-3333
0004	John	Smith	(555) 555-4444

Primary Keys

A **primary key** makes it possible to uniquely identify each row in a table.

To get any piece of information from a database, you should need only three things:

1) The name of the table;
2) The name of the column;
3) The primary key of the row.

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Primary Keys

A **primary key** makes it possible to uniquely identify each row in a table.

To get any piece of information from a database, you should need only three things:

1) The name of the table;
2) The name of the column;
3) The primary key of the row.

Example:

customer (customer_number, first_name, last_name)

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Primary Keys

Some notes about choosing/creating primary keys:

- Primary keys should be unique. **why?**
- Primary keys should not be null. **why?**

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Primary Keys

Some notes about choosing/creating primary keys:

- Primary keys should be immutable, meaning they can't change. (Can you think of why? Think back to our ER diagrams.)

- Be careful when creating primary keys that encode some meaning, as that data may change. It must be done in a very careful way.

Primary Keys: Concatenated

Some notes about choosing/creating primary keys (continued):

- If no single column fits the requirements of being immutable and unique, you can use two or more columns; this is called a **concatenated primary key** or **composite key.**

transaction (customer_id, time, item_id, comment)

- Is it okay for any of these columns to be null? Why or why not?

Primary Keys: Concatenated

Some notes about choosing/creating primary keys (continued):

- If no single column fits the requirements of being immutable and unique, you can use two or more columns; this is called a **concatenated primary key** or **composite key.**

transaction (customer_id, time, item_id, comment)

One should use as few columns as possible, and the chosen columns should be those that will never change; meaningless identifiers are best.

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Primary Keys: Surrogate

What if there really is no attribute or combination of attributes that could serve as a primary key? For example, consider the following relation:

customer (first_name, last_name , address)

It's unlikely that someone with the same first and last name would live at the same address, but it's possible! (and probably happens more often than you think.)

Primary Keys: Surrogate

customer (customer_id, first_name, last_name , address)

In this case, we have no choice but to **make up a new attribute** for a unique ID number (called **customer_id** in the above relation). This is sometimes called a **surrogate key.** It's one more column to keep track of and one more bit of complexity in the schema, so we often use it as a last resort.

Representing Data Relationships

As we've seen, we can represent a relationship between two tables by having a column in one table that contains primary key values from the other table. This column is called a **foreign key**.

What is the foreign key in the following example?

```
customer (customer_numb, first_name, last_name, phone)

order (order_numb, customer_numb, order_date)
```

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Representing Data Relationships

As we've seen, we can represent a relationship between two tables by having a column in one table that contains primary key values from the other table. This column is called a **foreign key**.

What is the foreign key in the following example?

customer (customer_numb, first_name, last_name, phone)

order (order_numb, customer_numb, order_date)

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Representing Data Relationships

A foreign key may form part of the primary key. However, foreign keys can sometimes contain null values which generally aren't allowed in primary keys, so watch out!

Also, we must enforce **referential integrity**, which means that a non-null value of a foreign key ***must*** refer to an existing primary key somewhere. **(Why?)**