Concepts of the Relational Model

Relation Schemata

- An *attribute* is just a name.

- A *relation schema* is (formally) just a name R for the schema, together with a set **A** of attributes. Write R(**A**).

- Example:
- Let Name, ID_number, and Major be attributes. Then

  Student({Name, ID_number, Major})
- is a relation schema.

- Formally, there is no ordering of the attributes implied, but in practice one often writes with an (unofficial) ordering.

- Example:

  Student(Name, ID_number, Major)

- This might also be depicted as:

Student	Name	ID_number	Major

Instances of Relation Schemata

- A *domain* for an attribute is a set of values for the attribute. If A is an attribute, then $\mathfrak{D}(A)$ denotes the domain of A.

- Examples:

  $\mathfrak{D}(\text{ID_number})$ ={xxxxxx-xxxx | x is a digit}.

  $\mathfrak{D}(\text{Major})$ = {Computer Science, Computer Engineering, Business Data Processing}.

  $\mathfrak{D}(\text{Name})$ = String of characters.

- A *tuple* over the set $\mathbf{A}$ of attributes is a function f on the domain $\mathbf{A}$ such that for each $A \in \mathbf{A}$, $f(A) \in \mathfrak{D}(\mathbf{A})$.

- The set of all tuples over $\mathbf{A}$ is denoted Tuple($\mathbf{A}$).

- Example: f operates as follows:

Name	$\mapsto$	Kari Nordmann
ID_number	$\mapsto$	771030-0123
Major	$\mapsto$	Computer Engineering

- This notation becomes very awkward in a hurry. If we adopt an ordering convention for the attributes, such as (Name, ID_number, Major), then we may write this tuple much more succinctly as

(Kari Nordmann, 771030-0123, Computer Engineering)

- Sometimes, null values are allowed in the range of the function f as well.

(Kari Nordmann, 771030-0123, NULL)

- A *relation instance* r for a relation schema is a set of tuples over its attribute set. The set of all relation instances for R[**A**] is denoted $\mathfrak{I}$(R[**A**]).

Example:

For the running example, here is an instance, expressed in a more usual notation.

Student	Name	Major	ID_number
	Kari Nordmann	CE	771030-0123
	Ola Nordmann	CS	721225-0134
	Bill Smith	BDP	600101-0554
	Jane Smith	BDP	600704-0144
	Renée Française	CE	650501-0164

It is important to know that the order in which the tuples are presented is of no special importance. The following represents exactly the same instance.

Student	Name	Major	ID_number
	Renée Française	CE	650501-0164
	Bill Smith	BDP	600101-0554
	Kari Nordmann	CE	771030-0123
	Jane Smith	BDP	600704-0144
	Ola Nordmann	CS	721225-0134

First Normal Form

- There is actually a small flaw in the previous design. The name field is compound, in that it contains both the first and the last name of the student. If it is desired to extract these parts of the total name, then this arrangement is unacceptable in the relational model. Formally:

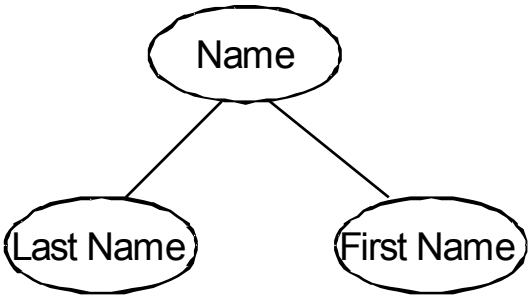
- A relation schema is in *first normal form* if each of the domains of its attributes is *atomic* in the sense that these domain elements cannot, for the purposes of the model, be decomposed further.

- It is a fundamental requirement that a relational design be in first normal form.

- To place the previous design in first normal form, something like the following is needed.

Student	Last Name	First Name	Major	ID_number
	Nordmann	Kari	CE	771030-0123
	Nordmann	Ola	CS	771225-0134
	Smith	Bill	BDP	600101-0554
	Smith	Jane	BDP	600704-0144
	Française	Renée	CE	600501-0164

With this solution, however, it is no longer possible to refer to the name as a unit, as embodied in the following ER diagram.



More will be said later about mapping ER representations to the relational model.

Key Constraints on Relation Schemata

- For a relation schema R[**A**], a constraint C is just a subset of the set of all relation instances for R[**A**].
- The set of all instances which satisfy C is denoted Sat(R[**A**],C).
- A relation instance r is said to *satisfy* constraint C if $r \in$ Sat(R[**A**],C).

- Important: A constraint is a property of the set of allowable relations. It is not a property of a particular relation.

- Let $\mathbf{B} \subseteq \mathbf{A}$, and let $r \in \Im(R[\mathbf{A}])$. It is said that r satisfies the constraint Superkey(R[**A**],**B**) if, whenever $t_1, t_2 \in$Tuple(**A**), it is the case that

$$t_1[\mathbf{B}] = t_2[\mathbf{B}] \Rightarrow t_1 = t_2.$$

- In this case, **B** is called a *superkey* of R[**A**].

- If **B** is a superkey, and it is the case that there is no proper subset of **B** which is also a superkey, then **B** is called a *candidate key*, or sometimes just a *key*.

Primary Keys

- In general, there may be many candidate keys for a relation under a constraint set C. Usually, a particular candidate key is designated as the *primary key.* The attributes of the primary key are underlined in many notations.

Example:

Student	Last Name	First Name	Major	ID_number

- Most systems insist that each relation have a primary key.

Relational Database Schemata

- Informally, a *relational database schema* is a collection of relation schemata, together with some constraints on their values.

Example:

Student	Major	ID_number

Student_Name	ID_number	Last Name	First Name

- To proceed formally, some further definitions are needed.

Formalization of Relational Database Schemata

- A *free relational database schema* $\mathcal{R}$ is a set of relation schemata.

- An *instance* of $\mathcal{R}$ is just a collection $\mathcal{I}$ of relation instances, one for each relation schema in $\mathcal{R}$. The set of all instances of $\mathcal{R}$ is denoted $\mathfrak{I}(\mathcal{R})$, and the relation associated with $R \in \mathcal{R}$ for instance $\mathcal{I}$ is denoted $R^{\mathcal{I}}$.

A *constraint* on $\mathcal{R}$ is a subset of the set of all instances of $\mathcal{R}$.

- A *relational database schema* is a pair $(\mathcal{R},C)$ in which $\mathcal{R}$ is a free relational database schema and C is a set of constraints on $\mathcal{R}$.

- A constraint on a relation scheme $R[\mathbf{A}] \in \mathcal{R}$ is interpreted as a constraint on $\mathcal{R}$ in the obvious way.

  Example: The relational database schema shown on the previous slide has two primary-key constraints.

Entity and Foreign-Key Constraints

- According to the text, an *entity integrity constraint* asserts that a primary key may not be null. It will always be assumed that this condition is implicit in the declaration of a primary key.

- Let $\mathcal{R}$ be a free relational database schema, and let $R_1[\mathbf{A}_1]$ and $R_2[\mathbf{A}_2] \in \mathcal{R}$. Let $\mathbf{F} \subseteq \mathbf{A}_2$, and let $\mathbf{K} \subseteq \mathbf{A}_1$ be the primary key of R_1. An instance $\mathcal{I} \in \mathfrak{I}(\mathcal{R})$ satisfies the *foreign-key constraint*

  ForeignKey($\mathcal{R}$, $R_1[\mathbf{A}_1]$, $R_2[\mathbf{A}_2]$, **F)**

  if the following conditions are satisfied:

- There is a bijection $k : \mathbf{K} \rightarrow \mathbf{F}$ with the following properties:

  - $\mathfrak{D}(A_i) = \mathfrak{D}(k(A_i))$ for each $A_i \in \mathbf{K}$.

  - For each tuple $t_2 \in R_2^{\mathcal{I}}$, either every attribute in **F** has a null value or else there is a tuple $t_1 \in R_1^{\mathcal{I}}$ such that $t_1[\mathbf{K}] = t_2[\mathbf{F}]$.

  In this case, it is said that **F** is a *foreign key* for R_2.

  Example: Figure 5.7 of the text. (7.7 in the Third Edition)

Other Types of Constraints

Over the years, many different forms of integrity constraints have been proposed for relational database systems. However, other than the types mentioned above, few have been implemented. The reasons:

- The computational complexity of checking the validity of these constraints is too high.
- The constraints are too specialized.

General Comment:

The relational model is closely tied to first-order logic. This makes it highly amenable to theoretical research, of which there has been a great deal over the past thirty-five years.