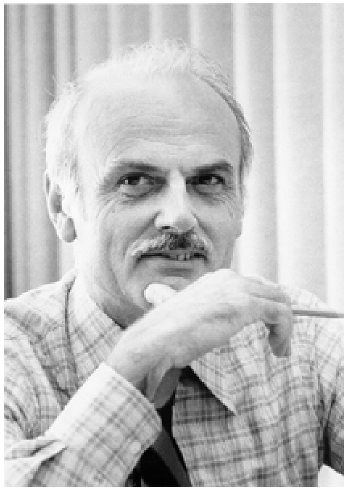
Relational Data model

Relational Model

introduced by Frank Edgar Codd (1923-2003)

- in the classic paper “A Relational Model for Large Shared Data Banks” in 1970
- Codd worked in IBM Almaden Research Center, St. Jose
- ACM Turing Award in 1981 for ”fundamental and continuing contributions to the theory and practice of database systems“



Relational model:

- uses the concept of mathematical relation
- theoretical basis in the **set theory** and **first-order predicate logic**

To put it simple:

- Storing database in simple data structures (tables)
- Access data through a high-level language

Relational model concepts

Database represented as a collection of **relations**

- a table is called a **relation**
- a raw is called a **tuple**
- a column is called an **attribute**
- a data type of possible values in a column is called a **domain**

Domain

Domain *D* is a set of atomic values

– atomic values are indivisible

– data types can be strings, integers, reals

– domains can also be specific set of values

  – zip codes
  – date of birth (dd-dd-dddd) with specific constraints on these digits
  – age (can be a natural number (integer) or zero)

Relation schema, domain, attribute

A **relation schema** *R* denoted by $R = (A_1, A_2, \ldots, A_n)$ is made of

- relational name *R* and
- a list of attributes $A_1, A_2, \ldots, A_n$

An **attribute** A_i is a role played by some domain D denoted by $dom(A_i)$

The **degree** (or **arity**) of a relation is the number of attributes n

Example:

- *STUDENT (SSN, Name, Home_phone, Age, Address)*
- *STUDENT (SSN:string, Name:string, Home_phone:string, Age:integer, Address:string)*
  - here with specified domains

Relation (relation state)

A **relation** (**relation state**) r of a relation schema $R = (A_1, A_2, \ldots, A_n)$ denoted as $r(R)$ is a **set** of tuples $r = t_1, t_2, \ldots, t_m$

- tuple is an ordered list $t =< v_1, v_2, \ldots, v_n >$ of n values
- $v_i \in dom(A_i)$ or v_i $is\ NULL$ as a special value for each $i \in 1, 2, \ldots, n$

The corresponding terms are used:

- *relation intension* - for relation schema R
- *relation extension* - for relation state $r(R)$

The definition of the mathematical set has the following features:

- it does not have duplicate elements
- there is no order among it's elements

Binary relation - formal definition

Given two sets A and B. A binary **relation** over sets A and B is a *subset* of the *Cartesian product* A and B.

$$R \subseteq \{A \times B\}$$

Binary relation is an unordered set of ordered pairs.

$$R \subseteq \{(a_1, b_1), \ldots, (a_k, b_k)\}$$

n-ary relation

If $A_1, A_2, \ldots, A_n$ are sets. An n-ary **relation** over sets $A_1, A_2, \ldots, A_n$ is the *subset* of the *n-ary Cartesian product* of those sets.

$$R \subseteq \{A_1 \times A_2 \times \ldots \times A_n\}$$

In relational algebra n-ary relation is called just *relation* or *relation state* and we have:

$$r(R) \subseteq (dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$$

- total number of possible tuples (rows), or **cardinality** of relation R is $|dom(A_1)| \times |dom(A_2)| \times \ldots \times |dom(A_n)|$ where $|dom(A_i)|$ is the cardinality of set A_i
- in mathematics, a relation is a set of tuples which does not have any particular order
  - by contrast, in a file (database), tuples are physically ordered and duplicates might exist

Interpretation of relation in the Relational Model

Facts and relationships are both represented in relations. According to the relational model, **relationships** between entities are integrated in relations.

- some attributes contain values of attributes of the other relations and that is how relationships are represented
- entities and relationships are represented *uniformly*

Closed world assumption holds in the relation model

- facts in the database are *the only* true facts in the universe

Null values

NULL value has several meanings

- value unknown
- value exists but is not available
- attribute does not apply (value undefined)

Behavior of NULL values in comparisons and aggregations

- if both A an B have NULL values, it doesn’t mean they represent the same values when compared to each other

Behavior of Nulls and logical AND

Boolean logic truth table for the logical conjunction (logical and) with nulls:

A	**B**	$A \wedge B$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$
$\top$	null	null
null	$\top$	null
$\bot$	null	$\bot$
null	$\bot$	$\bot$
null	null	null

The logical expression with AND can never be true if one of the operands is “undefined” (null).

- true and null gives an unknown value, i.e., null
- however, false and null gives false value

Behavior of Nulls and logical OR

Boolean logic truth table for the logical disjunction (logical operator **OR**) with nulls:

A	**B**	$A \vee B$
$\top$	$\top$	$\top$
$\top$	$\perp$	$\top$
$\perp$	$\top$	$\top$
$\perp$	$\perp$	$\perp$
$\top$	null	$\top$
null	$\top$	$\top$
$\perp$	null	null
null	$\perp$	null
null	null	null

The logical expression OR can never be false if one of operands is “undefined” (null).

– false and null gives an unknown value

– however, null or true gives always true

Key Constraints

Because relation is defined as a set of tuples it holds

– all tuples in a relation must be distinct which can be denoted as

$$t_i[R] \neq t_j[R], \ \forall i, j, i \neq j$$

Usually there are other subsets of attributes with the property that no two tuples in any relation state r of R should have the same combination of values for these attributes

Superkey

Def: Let SK be a set of attributes in a relation schema R. If for any two distinct tuples t_i and t_j in a relation state r of R holds

$$t_i[SK] \neq t_j[SK]$$

then such set of attributes is called a **superkey**.

The Superkey SK specifies a *uniqueness constraint*

– one relation can have many superkeys

– every relation has at least one superkey
  What is that superkey?

– a superkey can have redundant attributes.

Key

We are interested for those superkeys which do not have redundant attributes.

Def: A **key** *K* of a relation schema R is a superkey of R with the additional property that removing any attribute *A* from *K* leaves a set of attributes K' that is not a superkey of R any more.

Key satisfies two conditions:

1. no two tuples in any state of the relation can have identical values of all attributes in the key
2. it is a *minimal superkey* - removing any attributes from it violates the uniqueness constraint in condition 1.

A key is also a superkey but not vice versa

Primary key and candidate keys

A relation schema may have more keys and they are called **candidate keys**.

Def:A **primary key** is one of candidate keys that is chosen among others and used to identify tuples in the relation.

- we denote a primary key as an underlined set of attributes
- other candidate keys which are not the primary key are called **unique keys**

Student

<u>SSN</u>	MATR_NUM	STUDENT_NAME	CLASS
123-45-6789	9240006	John Brown	1
050-42-3729	5765763	Christine Smith	2
527-42-1289	1069362	Leslie Connor	1
103-42-4789	2795741	John Viener	1
416-41-1298	3761763	Leslie Connor	3

Name all possible candidate keys in the *Student* relation?

Primary keys and candidate keys

Relation: STUDENT(SSN, MATR_NUM, STUDENT_NAME,CLASS)

Superkeys:

- SSN, MATR_NUM, STUDENT_NAME, CLASS
- SSN, MATR_NUM, STUDENT_NAME
- SSN, MATR_NUM, CLASS
- SSN, STUDENT_NAME, CLASS
- MATR_NUM, STUDENT_NAME, CLASS
- SSN, STUDENT_NAME
- SSN, CLASS
- MATR_NUM, STUDENT_NAME
- MATR_NUM, CLASS
- SSN, MATR_NUM

Candidate keys

- SSN
- MATR_NUM

Primary key

- SSN

Foreign keys

Def: A set of attributes *FK* in relation schema R_1 is a **foreign key** of R_1 that references relation R_2 if it satisfies the following rules:

1. the attributes *FK* **refer to** the relation R_2
   attributes in *FK* have the same domains as the primary key in R_2
2. every tuple of R_1 **refers to** a tuple of R_2

$$t_1[FK] = t_2[PK]$$

   i.e value of FK in any tuple t_1 in the current state $r(R_1)$ is either some value of some tuple t_2 in the current state $r(R_2)$, or it is NULL

This constraint on relation schema is called **referential integrity constraint**

A foreign key can also refer to the primary key of its own relation

Foreign keys example

CUSTOMER

SSN	CUSTOMER_NAME	ADDRESS
123-45-6789	John Brown	...
050-42-3729	Christine Smith	...
527-42-1289	Leslie Connor	...
103-42-4789	Erika Viener	...
416-41-1298	Leslie Connor	...

ORDER

ORDER_ID	PRODUCT_NAME	CUSTOMER_ID
9240006	Spicy Pizza Balado	123-45-6789
5765763	Shakey's Pizza	050-42-3729
1069362	Tandoori Paneer	050-42-3729
2795741	Tandoori Paneer	NULL
3761763	Tandoori Paneer	123-45-6789

foreign key (CUSTOMER_ID) references CUSTOMER(SSN)

- note - second relation has a wrong design decision

Functional dependency constraint

Given two sets of attributes X and Y. A **functional dependency** denoted by $X \rightarrow Y$, holds if :

$$\forall t_i, t_j \in r(R) : t_i[X] = t_j[X] \implies t_i[Y] = t_j[Y]$$

– values of X uniquely determine values of Y

In common parlance:

– Y is functionally dependent on X

– if two tuples agree on X values, they must necessarily agree on their Y values

Functional dependency example

Student

SSN	MATR_NUM	STUDENT_NAME	CLASS
123-45-6789	9240006	John Brown	1
050-42-3729	5765763	Christine Smith	2
527-42-1289	1069362	Leslie Connor	1
103-42-4789	2795741	John Viener	1
416-41-1298	3761763	Leslie Connor	3

The following holds:

- $SSN \rightarrow \{SSN, MATR\_NUM, STUDENT\_NAME, CLASS\}$
- $SSN \rightarrow \{MATR\_NUM, STUDENT\_NAME\}$
- $\{SSN, MATR\_NUM\} \rightarrow \{MATR\_NUM, STUDENT\_NAME\}$
- $\{MATR\_NUM\} \rightarrow \{STUDENT\_NAME, CLASS\}$
- $\ldots$

Functional dependency inference rules (Armstrong's axioms)

- Reflexivity: $Y \subseteq X \implies X \rightarrow Y$
- Augmentation: $X \rightarrow Y \implies XZ \rightarrow YZ$
- Transitivity: $X \rightarrow Y \wedge Y \rightarrow Z \implies X \rightarrow Z$
- Decomposition: $X \rightarrow YZ \implies X \rightarrow Y \wedge X \rightarrow Z$
- Union: $X \rightarrow Y \wedge X \rightarrow Z \implies X \rightarrow YZ$
- Pseudotransitivity: $X \rightarrow Y \wedge WY \rightarrow Z \implies WX \rightarrow Z$

Comutativity doesn't hold

Relational database schema

Def: A **relational database schema** S is a set of relation schemas

$$S = \{R_1, R_2, \dots, R_n\}$$

and a set of **integrity constraints** IC

Def: A **relational database state** (or **relational database instance**) of S is a set of relation states

$$DB = \{r_1, r_2, \dots, r_m\}$$

such that r_i is a relation state of R_i and such that relation states satisfy all constraints in IC.

Operations of the relational model

Operations of the relational model can be categorized as retrievals and updates

- retrievals are explained in the relational algebra
- update operations
  ~~1.~~ Insert operations
  ~~2.~~ Update operation
  ~~3.~~ Delete operation

Review questions

– Explain the notation of relation and its components?
– How are the terms of functional dependency and key related?
– What is the difference between a key and a superkey?
– Clarify the difference between a relation intension and a relation extension?