Introduction to Relational Model
Chapter - 2

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan

Example of a Relation

attributes
(or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples
(or rows)

©Silberschatz, Korth and Sudarshan

A Sample Relational Database

instructor

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

department

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

©Silberschatz, Korth and Sudarshan

Attribute Types

- Each attribute of a relation has a name
- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**, that is, indivisible
  - E.g. multivalued attribute values are not atomic
  - E.g. composite attribute values are not atomic
- The special value *null* is a member of every domain
- The null value causes complications in the definition of many operations

Relation Schema

- $A_1, A_2, \ldots, A_n$ are *attributes*

- $R = (A_1, A_2, \ldots, A_n)$ is a **relation schema**

  E.g.

  instructor = (*ID, name, dept_name, salary*)

  Customer-schema =

  (*customer-name, customer-street, customer-city*)

- Formally, given sets $D_1, D_2, \ldots D_n$ a **relation *r*** is a subset of

  $D_1 \times D_2 \times \ldots \times D_n$

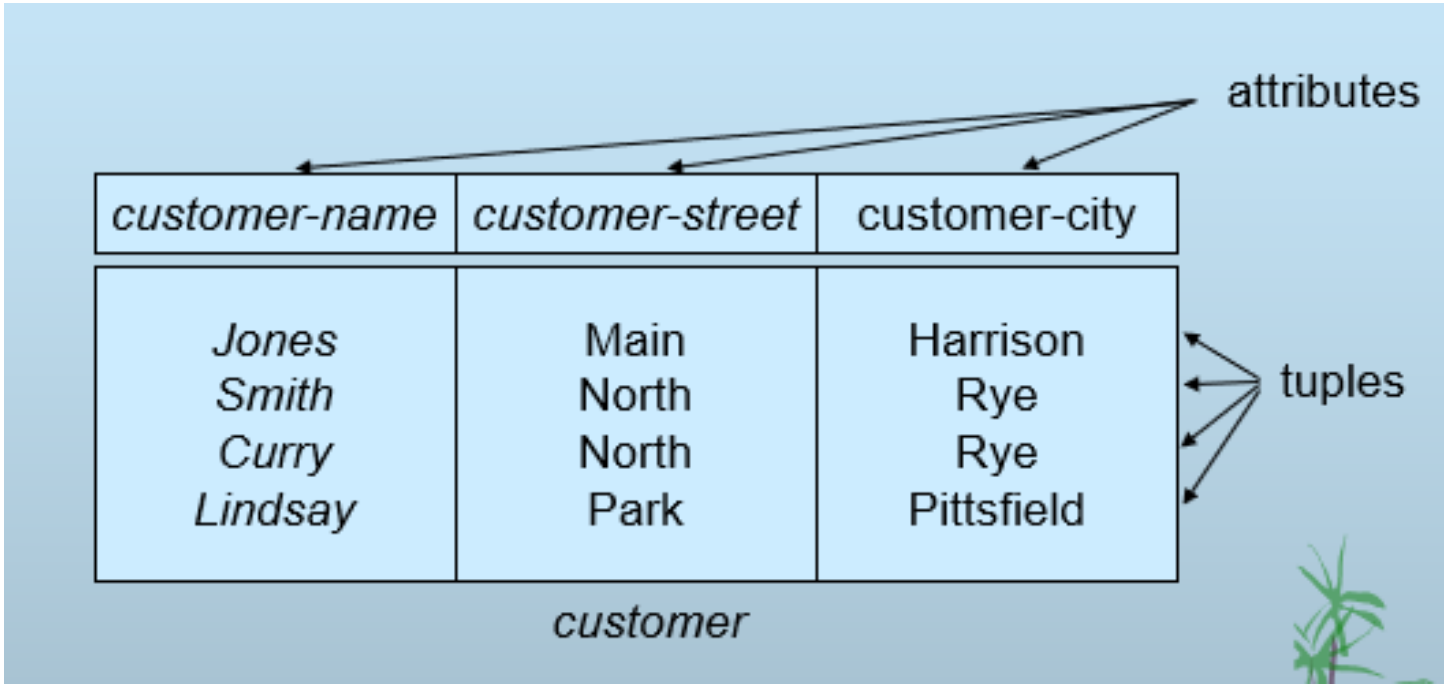
  Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$

- $r(R)$ is a *relation* on the *relation schema R*

  E.g. *customer (Customer-schema)*

Relation Instance

- The current values (**relation instance**) of a relation are specified by a table
- An element ***t*** of ***r*** is a *tuple*, represented by a *row* in a table



Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

©Silberschatz, Korth and Sudarshan

Database

- A database consists of multiple relations
- Information about an enterprise is broken up into parts

  instructor - ID, name , salary, department
  student - ID, name, program, credits, advisor
  advisor - ID, name, topic

- Bad design:

  university (*instructor_ID, name, dept_name, salary, student_ID*, …)

  results in

  - repetition of information (e.g., two students have the same instructor)
  - the need for null values (e.g., represent an student with no advisor)

- Normalization theory deals with how to design relational schemas

Keys

- Let K $\subseteq$ R
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*
- Superkey *K* is a **candidate key** if *K* is minimal

  Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - which one?

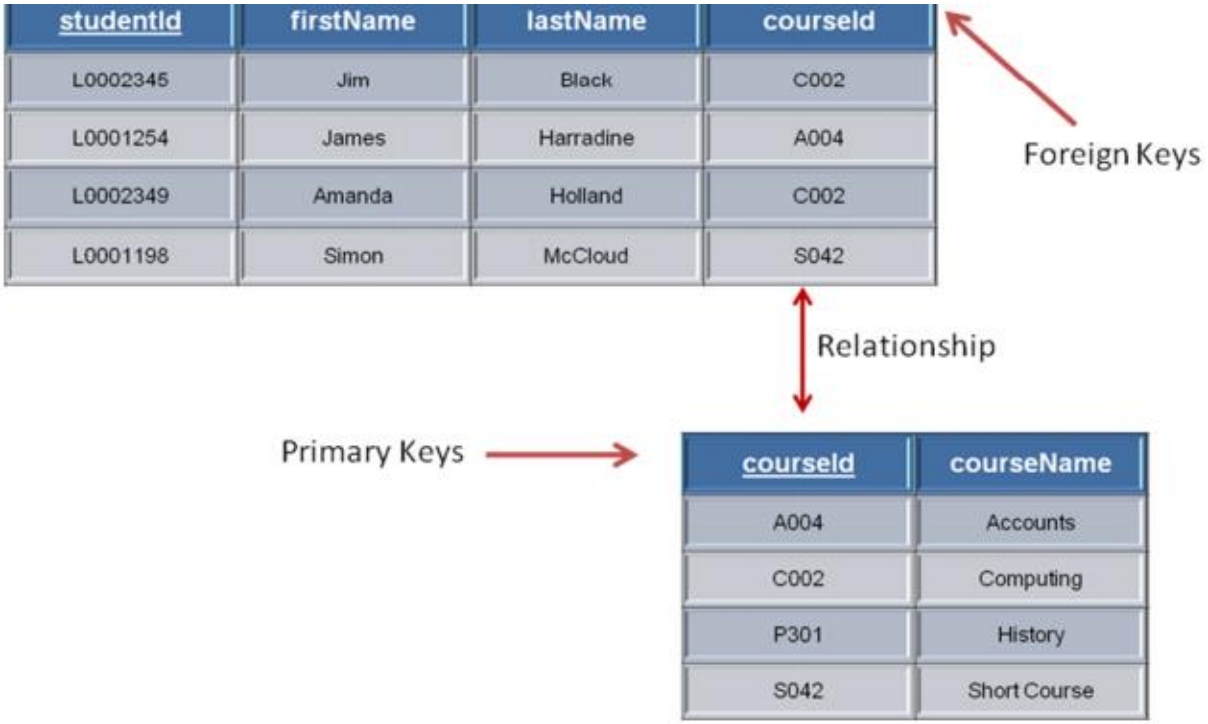
Primary Key

- **Primary key:** a candidate key chosen as the principal means of identifying tuples within a relation.
  - Should choose an attribute whose value never, or very rarely, changes.
  - E.g. *email address* is unique, but may change
  - When choosing a primary key from the pool of candidate keys always choose a single simple key over a composite key.
- Examples:
  - *department*(*dept name*, *building*, *budget*)
  - *classroom*(*building*, *room number*, *capacity*)
  - *course*(*course id*, *title*, *dept name*, *credits*)
  - *instructor*(*ID*, *name*, *dept name*, *salary*)
  - *student*(*ID*, *name*, *dept name*, *tot cred*)

©Silberschatz, Korth and Sudarshan

Foreign Key

- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation, **Referenced** relation
  - E.g. *dept_name* in i*nstructor* is a foreign key from *instructor* referencing *department*
- A **foreign key** is generally a primary key from one table that appears as a field in another where the first table has a relationship to the second.



Referential Integrity

- Ensures that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.
  - Example: If "Biology" is a department name appearing in one of the tuples in the *instructor* relation, then there exists a tuple in the *department* relation for "Biology".
- Let A be a set of attributes. Let R and S be two relations that contain attributes A and where A is the primary key of S. A is said to be a **foreign key** of R if for any values of A appearing in R these values also appear in S.

©Silberschatz, Korth and Sudarshan

Cascading Actions in Referential Integrity

- **create table** *course (*
  course_id **char**(5) **primary key**,
  title **varchar**(20),
  dept_name **varchar**(20) **references** *department*
)
- **create table** *course (*
  …
  dept_name **varchar**(20),
  foreign key (*dept_name*) **references** *department*
  on delete cascade
  on update cascade,
  . . .
)
- alternative actions to cascade: **set null**, **set default**

©Silberschatz, Korth and Sudarshan

Integrity Constraint Violation

- E.g.

  create table *person* (
  ID **char**(10),
  name **char**(40),
  mother **char**(10),
  father **char**(10),
  primary key *ID*,
  foreign key *father* **references** *person*,
  foreign key *mother* **references** *person*)

- How to insert a tuple without causing constraint violation ?
  - insert father and mother of a person before inserting person
  - OR, set father and mother to null initially, update after inserting all persons (not possible if father and mother attributes declared to be **not null**)

Schema Diagram for University Database

A database schema, along with primary key and foreign key dependencies, can be depicted by **schema diagrams**.