

The Relational Model

- database consists of several **tables (relations)**
- columns in each table are named by **attributes**
- each attribute has an associated **domain**
(set of allowed values)
- data in each table consists of a set of **rows (tuples)** providing values for the attributes

Example

The diagram shows a table with the following structure:

- Relation name:** An arrow points from the label "Relation name" to the cell containing "STUDENT".
- Attributes:** An arrow points from the label "Attributes" to the header row of the table, which contains "Name", "SSN", "HomePhone", "Address", "OfficePhone", "Age", and "GPA".
- Tuples:** An arrow points from the label "Tuples" to the first column of the data rows, which contains the names of the students.

STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

Relation Schema

“type declaration”

- Relation name
- Set of attributes
- Domain of each attribute
- Integrity constraints

Example

CUSTOMER (Cust-id, Cust-name, Address, Phone#)
integer char strings 6-digits

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
- Sometimes, the special value ***null*** is considered a member of every domain

Relation Instance

An **instance** of a relation schema is the current content of the relation: a **set of rows** (tuples) over the attributes, with values from the attribute domains

<i>customer_name</i>	<i>customer_street</i>	<i>customer_city</i>
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

More on tuples

Notation:

- We refer to **component values** of a tuple t by $t(A_i) = v_i$ (the value of attribute A_i for tuple t).

also called **coordinates**

Similarly, $t(A_u, A_v, \dots, A_w)$ refers to the *subtuple* of t containing the values of attributes A_u, A_v, \dots, A_w , respectively.

Example

<i>customer_name</i>	<i>customer_street</i>	<i>customer_city</i>
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

$t = \langle \text{Smith, North, Rye} \rangle$

$t(\text{customer_name}) = \text{Smith}$

$t(\text{customer_street}) = \text{North}$

$t(\text{customer_city}) = \text{Rye}$

attributes and tuple values are generally assumed to be ordered

Relations are Unordered Sets

The tuples are *not* considered to be ordered, even though they appear to be so in the displayed tabular form.

<i>account_number</i>	<i>branch_name</i>	<i>balance</i>
A-101	Downtown	500
A-215	Mianus	700
A-102	Perryridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Redwood	700
A-217	Brighton	750

Database

- A database consists of multiple relations
- Information about an application is broken up into parts, with each relation storing one part of the information
 - account* : stores information about accounts
 - depositor* : stores information about which customer owns which account
 - customer* : stores information about customers
- Storing all information as a single relation such as *bank (account_number, balance, customer_name, ..)* is possible but not desirable:
results in repetition of information and the need for null values

Relational Integrity Constraints

- Constraints are *conditions* that must hold on *all* valid relation instances of a database
- Some common types of constraints:
 1. **Key** constraints
 2. **Entity integrity** constraints
 3. **Referential integrity** constraints

Key Constraints

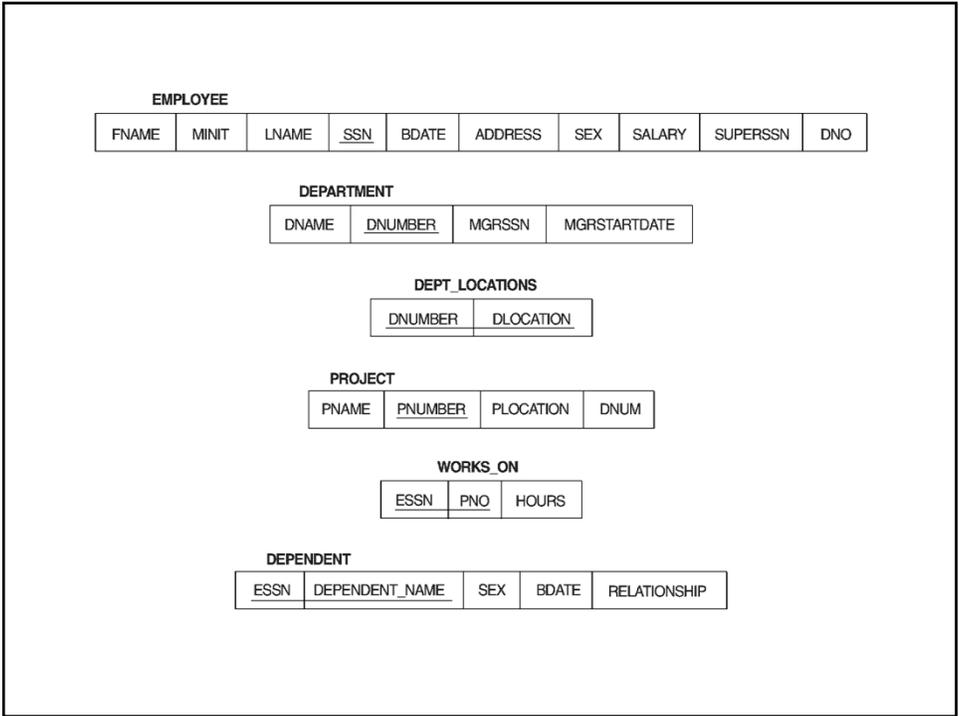
- **Superkey of R:** A set of attributes SK of R such that no two tuples *in any valid relation instance* $r(R)$ will have the same value for SK. That is, for all distinct tuples $t1$ and $t2$ in $r(R)$, $t1(SK) \neq t2(SK)$.
- **Key of R:** A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

Example: The CAR relation schema:
 CAR(State, Reg#, SerialNo, Make, Model, Year)
 has two keys Key1 = {State, Reg#}, Key2 = {SerialNo}.
 {SerialNo, Make} is a superkey but *not* a key.
- If a relation has *several* **candidate keys**, one is chosen arbitrarily to be the **primary key**.

Key Constraints

CAR	<u>LicenseNumber</u>	EngineSerialNumber	Make	Model	Year
	Texas ABC-739	A69352	Ford	Mustang	96
	Florida TVP-347	B43696	Oldsmobile	Cutlass	99
	New York MPO-22	X83554	Oldsmobile	Delta	95
	California 432-TFY	C43742	Mercedes	190-D	93
	California RSK-629	Y82935	Toyota	Camry	98
	Texas RSK-629	U028365	Jaguar	XJS	98

The primary key attributes are *underlined*.



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John			Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin			Wong	333445555	1965-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia			Zakya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer			Wallace	987654321	1941-06-20	291 Benny, Bellaire, TX	F	43000	888665555	4
Ramesh			Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce			English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad			Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James			Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1988-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

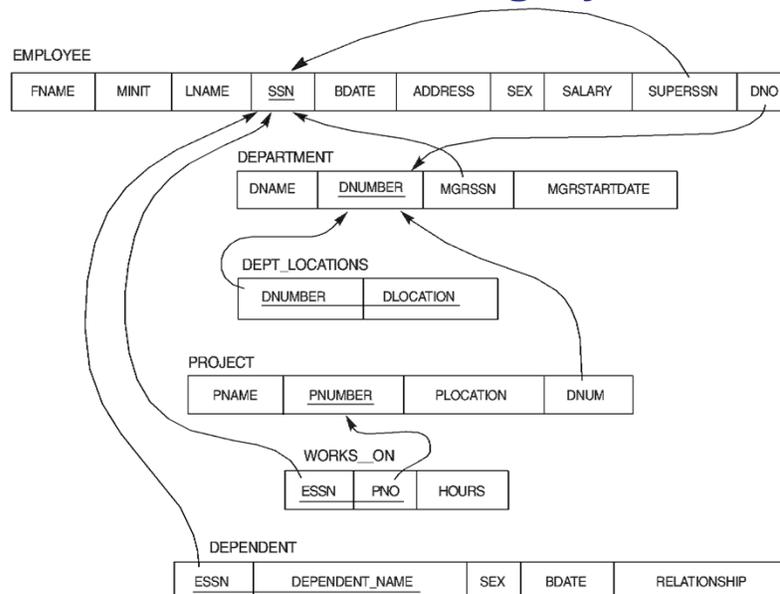
Entity Integrity

- The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple. This is because PK values are used to *identify* the individual tuples.

$t(A) \neq \text{null}$ for every tuple t in
a valid instance of R, where A is in PK

Note: Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity



Referential Integrity

- A constraint involving *two* relations of the database (the previous constraints involve a *single* relation).
- Used to specify a *relationship* among tuples in two relations: the **referencing relation** and the **referenced relation**.
- Tuples in the *referencing relation* R_1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation* R_2 . A tuple t_1 in R_1 is said to **reference** a tuple t_2 in R_2 if $t_1(\text{FK}) = t_2(\text{PK})$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1.\text{FK}$ to $R_2.\text{PK}$.

Referential Integrity Constraint

Statement of the constraint

The value in the foreign key column(s) FK of the **referencing relation** R_1 can be either (1) a value of a primary key PK in the **referenced relation** R_2 or (2) null.

In case (2), the FK in R_1 should not intersect its own primary key (or else entity integrity is violated)

Other Types of Constraints

- Semantic Integrity Constraints: based on application semantics and cannot be expressed by the model per se
 - e.g., “the max. no. of hours per employee for all projects he or she works on is 40 hrs per week”
 - A *constraint specification language* may have to be used to express these

SQL provides triggers and assertions

Update Operations on Relations

- INSERT a tuple.
- DELETE a tuple.
- MODIFY a tuple.

- Integrity constraints should not be violated by the update operations.
- Several update operations may have to be grouped together.

Update Operations on Relations

- In case of integrity violation, several actions can be taken:
 - Cancel the operation that causes the violation (REJECT option)
 - Perform the operation but inform the user of the violation
 - Trigger additional updates so the violation is corrected
 - Execute a user-specified error-correction routine