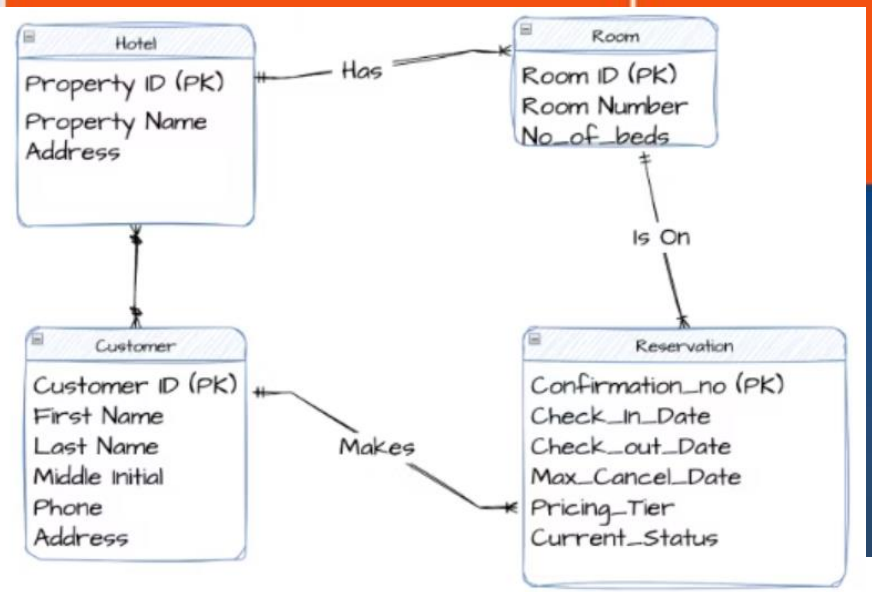
The Relational Data Model and Relational Database Constraints



The Relational Data Model and Relational Database Constraints

- Relational model
  - First commercial implementations available in early 1980s
  - Has been implemented in a large number of commercial system

A Logical View of Data

- Relational database model enables logical representation of the data and its relationships
- Logical simplicity yields simple and effective database design methodologies
- Facilitated by the creation of data relationships based on a logical construct called a relation

Informal Definitions

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows**.
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
  - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
  - In the formal model, the column header is called an **attribute name** (or just **attribute**)

Key of a Relation

- Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
  - Called the *key*
- Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
  - Called *artificial key* or *surrogate key*

Formal Definition — Schema

- The **Schema** (or description) of a Relation:
- Denoted by R(A1, A2, ...An)
- R is the **name** of the relation
- The **attributes** of the relation are A1, A2, ..., An
- Degree (or arity) is the number of attribute present in the relation.

CUSTOMER (Cust-id, Cust-name, Address, Phone#)

CUSTOMER is the relation name

Four attributes: Cust-id, Cust-name, Address, Phone#

Formal Definition — Tuple

- **tuple** is an ordered set of values (enclosed in angled brackets '< … >')
- Each value is derived from an appropriate *domain*.
- A relation is a **set** of such tuples (rows)
- Total number of tuples present in the table is called as cardinality

A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:
<632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
This is called a 4-tuple as it has 4 values
A tuple (row) in the CUSTOMER relation.

Formal Definition — Domain

- **Domain D** Set of atomic values, **Atomic** means Each value indivisible.
- A domain refers to the set of unique values used to define that attribute and will act as a "model" set of values.
- Theses values, being unique to that attribute are referred as **"atomic values"**
- It is a set of acceptable values that a column is allowed to store.

Example: attribute Cust-name is defined over the domain of character strings of maximum length 25
dom(Cust-name) is varchar(25)

Formal Definition — State

- The relation state is the actual content or data that is present in the table at a particular moment.
- The **relation state** is a subset of the Cartesian product of the domains of its attributes
- A table (the relation state) is just some of the possible combinations of the column values, and these possible combinations are defined by the domains of the attributes (columns).
- Each domain contains the set of all possible values the attribute can take.

Formal Definition — State

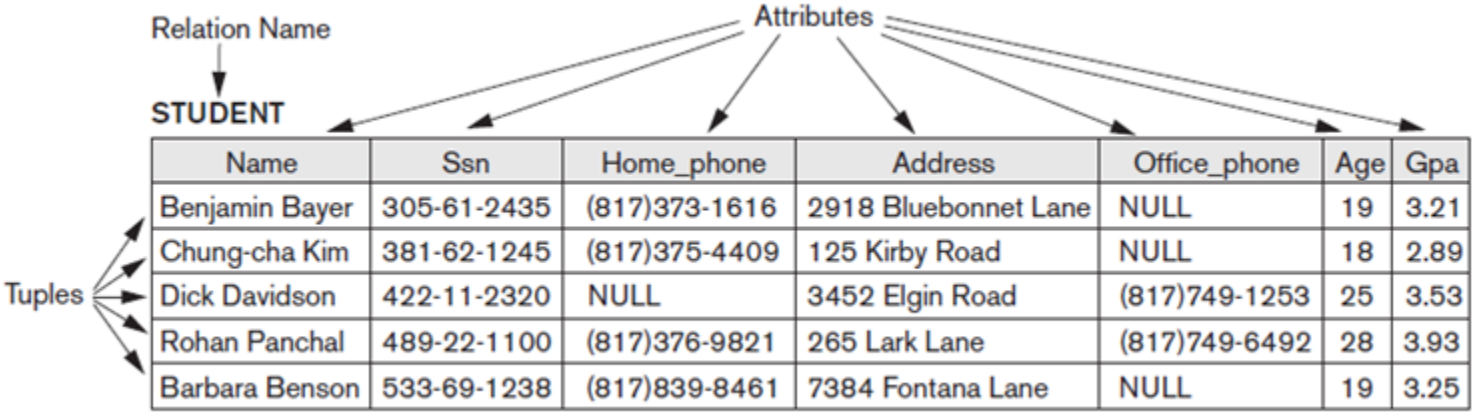
Cartesian Product:

If you take the domain of each attribute in a table and compute their Cartesian product, you get a set of all possible combinations of values for those attributes. For example, if a table has two attributes A and B with domains {1, 2} and {x, y}, respectively, then the Cartesian product would be the set of pairs {(1, x), (1, y), (2, x), (2, y)}.

Subset:

The actual data in the table (relation state) is a subset of this Cartesian product. This means that not every possible combination of attribute values is necessarily present in the table, but any combination that is present in the table must be one of those possible combinations from the Cartesian product.

Example — A Relation Student



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

RM Terminology

- **Relation**: table with columns and rows.
- **Attribute**: named column of a relation.
- **Domain**: set of allowable values for one or more attributes.
- **Tuple**: a record of a relation.
- **Relational Database** - collection of normalized relations with distinct relation names.

Formal Definition — Summary

Formally,

- Given R(A1, A2, ..., An)
- r(R) ⊂ dom (A1) X dom (A2) X ....X dom(An)

R(A1, A2, …, An) is the **schema** of the relation

R is the **name** of the relation

A1, A2, …, An are the **attributes** of the relation

r(R): a specific **state** (or "value" or “population”) of relation R – this is a *set of tuples* (rows)

- r(R) = {t1, t2, …, tn} where each ti is an n-tuple
- ti = <v1, v2, …, vn> where each vj *element-of* dom(Aj)

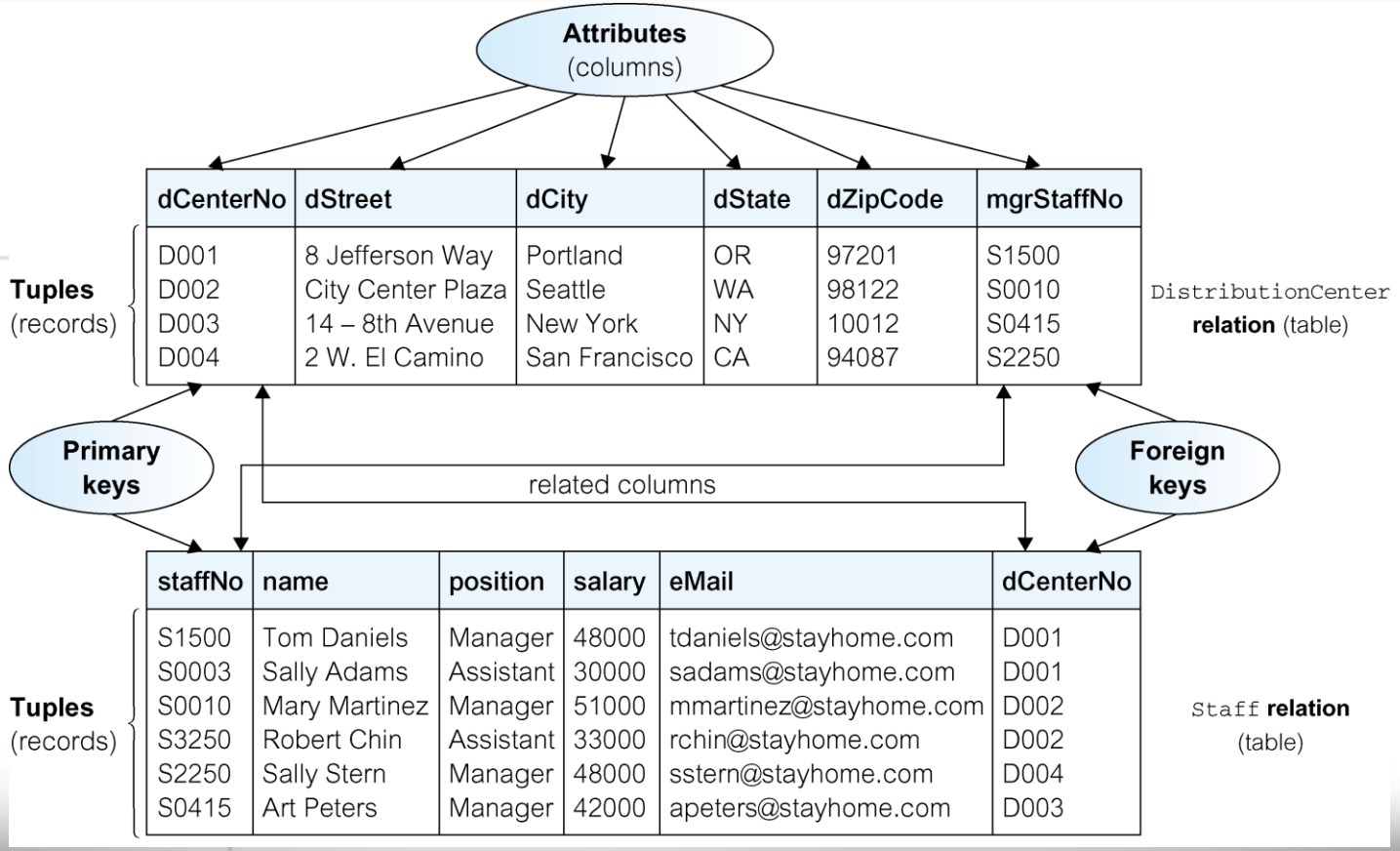
CHARACTERISTICS OF A RELATIONAL TABLE

1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (**tuple**) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute, and each column has a distinct name.
4	Each intersection of a row and column represents a single data value.
5	All values in a column must conform to the same data format.
6	Each column has a specific range of values known as the **attribute domain**.
7	The order of the rows and columns is immaterial to the DBMS.
8	Each table must have an attribute or combination of attributes that uniquely identifies each row.

Properties of Relational Tables

- Table name is distinct from all other table names in the database.
- Each cell of table contains exactly one atomic (single) value.
- Each column has a distinct name.
- Values of a column are all from the same domain.
- Each record is distinct; there are no duplicate records.
- Order of columns has no significance.
- Order of records has no significance, theoretically.

Instances of DistributionCenter and Staff relations



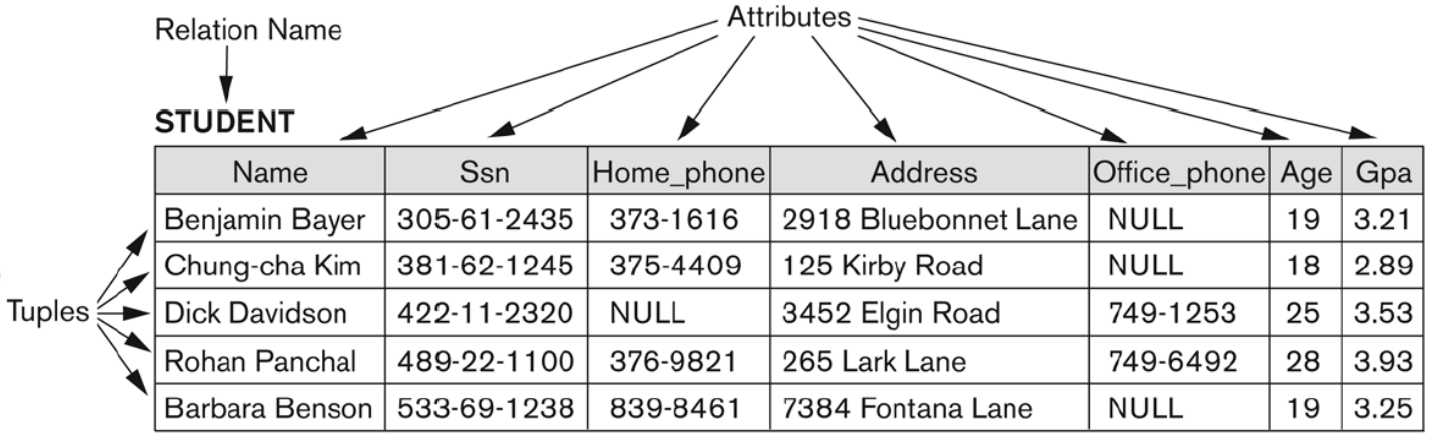
Domains for some attributes of distributionCenter and staff relations

Attribute	Domain name	Meaning	Domain definition
dCenterNo	DCenter_Numbers	Set of all possible distribution center numbers.	Character: size 4, range D001 – D999
dStreet	Street_Names	Set of all possible street names.	Character: size 60
staffNo	Staff_Numbers	Set of all possible staff numbers.	Character: size 5, range S0001 – S9999
position	Staff_Positions	Set of all possible staff positions.	Manager or Assistant
salary	Staff_Salaries	Possible values of staff salaries.	Monetary: 8 digits, range $10 000.00 – $100 000.00

Characteristics Of Relations

- **Ordering of tuples** in a relation r(R):
  - The tuples are *not considered to be ordered*, even though they appear to be in the tabular form.

Tuples Order



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	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
Rohan Panchal	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

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21

Characteristics Of Relations

- **Ordering of attributes** in a relation schema R (and of values within each tuple):
  - We will consider the attributes in R(A1, A2, ..., An) and the values in t=<v1, v2, ..., vn> to be ordered .

    (However, a more general alternative definition of relation does not require this ordering. It includes both the name and the value for each of the attributes).

    Example: t= { <name, “John” >, <SSN, 123456789> }

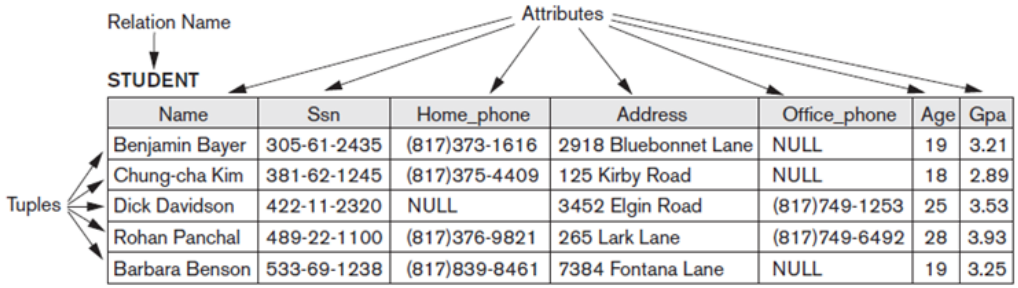
    This representation may be called as “self-describing”.

Characteristics Of Relations

- **Values in a tuple:**
  - All values are considered atomic (indivisible).
  - Each value in a tuple must be from the domain of the attribute for that column
    - If tuple t = <v1, v2, …, vn> is a tuple (row) in the relation state r of R(A1, A2, …, An)
    - Then each *vi* must be a value from *dom(Ai)*
  - A special **null** value is used to represent values that are unknown or not available or inapplicable in certain tuples.

Characteristics Of Relations

- **Meaning of a Relation:**
  - The relation schema can be interpreted as a declaration or a type of **assertion**.



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Student entitiy has a name,Ssn, Home_phone,address, Office_phone, Age,GPA.

  - Each tuple in the relation can be interpreted as **fact** or particular instance of the **assertion**

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three *main types* of (explicit schema-based) constraints that can be expressed in the relational model:
  - **Key** constraints
  - **Entity integrity** constraints
  - **Referential integrity** constraints
- Another schema-based constraint is the **domain** constraint
  - Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

CONSTRAINTS

Inherent or Implicit Constraints

- These are based on the data model itself. (E.g., relational model does not allow a list as a value for any attribute)

Schema-based or Explicit Constraints

- They are expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model)

Application based or semantic constraints

- These are beyond the expressive power of the model and must be specified and enforced by the application programs.

Relational Database Schema

- **Relational Database Schema:**
  - A set S of relation schemas that belong to the same database.
  - S is the name of the whole **database schema**
  - S = {R1, R2, ..., Rn} and a set IC of integrity constraints.
  - R1, R2, …, Rn are the names of the individual **relation schemas** within the database S

Relational Database State

- A **relational database state** DB of S is a set of relation states DB = $\{r_1, r_2, ..., r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy
- A relational database *state* is sometimes called a relational database *snapshot* or *instance*.
- A database state that does not meet the constraints is an invalid state

Populated database state

- Each *relation* will have many tuples in its current relation state
- The *relational database state* is a union of all the individual relation states
- Whenever the database is changed, a new state arises Basic operations for changing the database:
  - INSERT a new tuple in a relation
  - DELETE an existing tuple from a relation
  - MODIFY an attribute of an existing tuple

Populated database state for COMPANY

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

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

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	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

Key Constraints

- **Superkey** of R:
  - A **Superkey** is a set of one or more attributes (columns) in a relation (table) R that can uniquely identify each tuple (row) in that relation.
    - Is a set of attributes SK of R with the following condition:
      - No two tuples in any valid relation state r(R) will have the same value for SK
      - That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK]
      - This condition must hold in *any valid state* r(R)
- **Key** of R:
  - A **minimal superkey**, also known simply as a **key** or **candidate key**, is a superkey from which no attribute can be removed without losing the ability to uniquely identify each row in the table.

A super key can be the key(it its minimal) or may not be a key (if not minimal).

StudentID	Name	Email	Phone
101	Alice	alice@example.com	555-1234
102	Bob	bob@example.com	555-5678
103	Charlie	charlie@example.com	555-9012

Attributes in the Table:

- **StudentID**: A unique identifier for each student.
- **Name**: The student's name (not necessarily unique).
- **Email**: The student's email address (unique).
- **Phone**: The student's phone number (unique).

Possible Superkeys:

1. **{StudentID}**:
   - The StudentID is unique for each student, so it alone can uniquely identify each row in the table.
   - This is a superkey because no two rows have the same StudentID.
2. **{Email}**:
   - The Email is also unique for each student, so it can also uniquely identify each row in the table.
   - This is another superkey because no two rows have the same Email.
3. **{Phone}**:
   - The Phone number is unique for each student, making it another superkey.
4. **{StudentID, Email}**:
   - This combination can also uniquely identify each row because both StudentID and Email are unique.
   - Although it is a superkey, it is not minimal since StudentID or Email alone could do the job.
5. **{StudentID, Name}**:
   - Even though Name is not unique, when combined with StudentID, it uniquely identifies each row.
   - This is a superkey but not a minimal one because StudentID alone is sufficient.
6. **{StudentID, Email, Phone}**:
   - This is also a superkey since the combination of StudentID, Email, and Phone will uniquely identify each row.
   - However, this is far from minimal since each of these attributes on its own can be a superkey.

Candidate key

- Relation schema may have more than one key. Each of the keys is called the candidate key

StudentID	Name	Email	Phone
101	Alice	alice@example.com	555-1234
102	Bob	bob@example.com	555-5678
103	Charlie	charlie@example.com	555-9012

StudentID	Name	Email	Phone
101	Alice	alice@example.com	555-1234
102	Bob	bob@example.com	555-5678
103	Charlie	charlie@example.com	555-9012

Superkey Examples:

•**{StudentID, Email, Phone}**: This set of attributes can uniquely identify every row in the table. However, it's not minimal because you can remove Email and Phone and still have StudentID uniquely identify each row.

•**{StudentID, Email}**: This is also a superkey because the combination of StudentID and Email uniquely identifies every row. However, it's not minimal because StudentID alone is sufficient.

Minimal Superkey (Key):

•**{StudentID}**: This is a minimal superkey because:
- •It uniquely identifies each row (so it's a superkey).
- •If you remove StudentID, you lose the ability to uniquely identify the rows (meaning no other attribute or combination of attributes can do it on its own).

•{email}
•{Phone}

Key Constraints (continued)

- If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key**.
  - The primary key attributes are underlined.
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to *uniquely identify* each tuple in a relation
  - Provides the tuple identity
- Also used to *reference* the tuple from another tuple
  - General rule: Choose as primary key the smallest of the candidate keys (in terms of size)

CAR table with two candidate keys – LicenseNumber chosen as Primary Key

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

COMPANY Database Schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Entity Integrity

- **Entity Integrity:**
  - The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R).
    - This is because primary key values are used to *identify* the individual tuples.
    - If PK has several attributes, null is not allowed in any of these attributes

Referential Integrity

- A constraint involving **two** relations
  - The previous constraints involve a single relation.
- Used to specify a **relationship** among tuples in two relations:
  - The **referencing relation** and the **referenced relation**.
  - Dno is the foreign key of the employee, so employee is referencing relation and department is referenced relation.

Referential Integrity

- Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.
  - A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if t1[FK] = t2[PK].
  - A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Referential Integrity (or foreign key) Constraint

- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the the **referencing relation** R1 can be **either**:
    - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or
    - (2) a **null**.
- In case (2), the FK in R1 should **not** be a part of its own primary key.

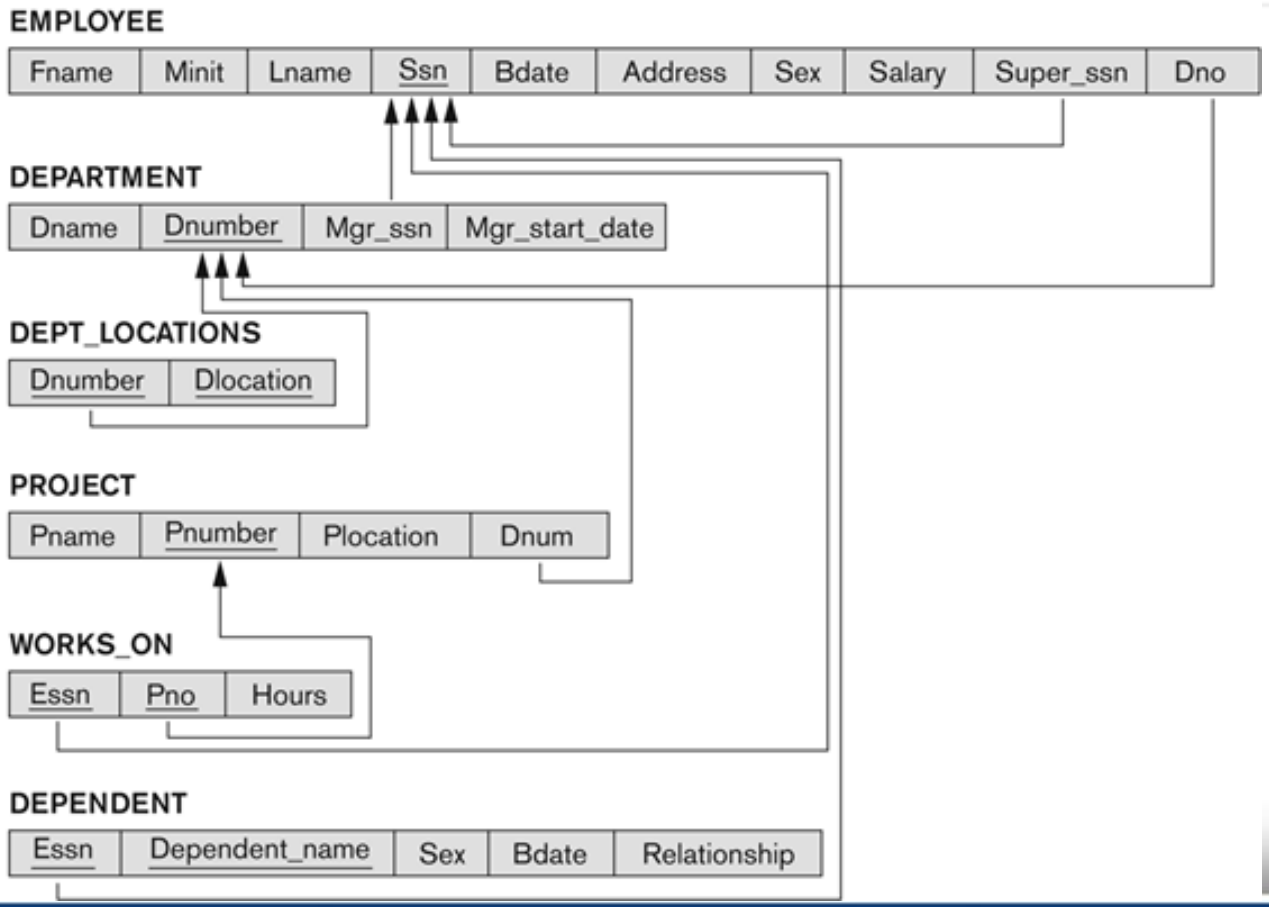
Department (Referenced Relation R2)	Employee (Referencing Relation R1)
DepartmentID (Primary Key)	**EmployeeID** (Primary Key)
DepartmentName	EmployeeName
	DepartmentID (Foreign Key)

The foreign key in the Employee table must either match an existing department's primary key in the Department table or be NULL. If it's NULL, it can't be part of the Employee table's primary key.

Displaying a relational database schema and its constraints

- Each relation schema can be displayed as a row of attribute names
- The name of the relation is written above the attribute names
- The primary key attribute (or attributes) will be underlined
- A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table
  - Can also point the the primary key of the referenced relation for clarity

Referential Integrity Constraints for COMPANY database in relation schema



RELATIONAL DATABASE KEYS

KEY TYPE	DEFINITION
Superkey	An attribute or combination of attributes that uniquely identifies each row in a table
Candidate key	A minimal (irreducible) superkey; a superkey that does not contain a subset of attributes that is itself a superkey
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row; cannot contain null entries
Foreign key	An attribute or combination of attributes in one table whose values must either match the primary key in another table or be null

Other Types of Constraints

- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the model per se
  - Example: “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- A **constraint specification** language may have to be used to express these
- SQL-99 allows **CREATE TRIGGER** and **CREATE ASSERTION** to express some of these semantic constraints
- Keys, Permissibility of Null values, Candidate Keys (Unique in SQL), Foreign Keys, Referential Integrity etc. are expressed by the **CREATE TABLE** statement in SQL.

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.
- Updates may **propagate** to cause other updates automatically. This may be necessary to maintain integrity constraints.

Update Operations on Relations

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

Possible violations for each operation

- INSERT may violate any of the constraints:
  - Domain constraint:
    - if one of the attribute values provided for the new tuple is not of the specified attribute domain
  - Key constraint:
    - if the value of a key attribute in the new tuple already exists in another tuple in the relation
  - Referential integrity:
    - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
  - Entity integrity:
    - if the primary key value is null in the new tuple

Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.

Result: This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected

Insert <'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.

Result: This insertion violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.

Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> into EMPLOYEE.

Result: This insertion violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.

Possible violations for each operation

- DELETE may violate only referential integrity:
  - If the primary key value of the tuple being deleted is referenced from other tuples in the database
    - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL
      - RESTRICT option: reject the deletion
      - CASCADE option: propagate the new primary key value into the foreign keys of the referencing tuples
      - SET NULL option: set the foreign keys of the referencing tuples to NULL
  - One of the above options must be specified during database design for each foreign key constraint

Delete the WORKS_ON tuple with Essn = '999887777' and Pno = 10.

Result: This deletion is acceptable and deletes exactly one tuple.

Delete the EMPLOYEE tuple with Ssn = '999887777'.

Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

Delete the EMPLOYEE tuple with Ssn = '333445555'.

Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations.

Possible violations for each operation

- UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified
- Any of the other constraints may also be violated, depending on the attribute being updated:
  - Updating the primary key (PK):
    - Similar to a DELETE followed by an INSERT
    - Need to specify similar options to DELETE
  - Updating a foreign key (FK):
    - May violate referential integrity
  - Updating an ordinary attribute (neither PK nor FK):
    - Can only violate domain constraints

Update the salary of the EMPLOYEE tuple with Ssn = '999887777' to 28000.

Result: Acceptable.

Update the Dno of the EMPLOYEE tuple with Ssn = '999887777' to 7.

Result: Unacceptable, because it violates referential integrity.

Update the Ssn of the EMPLOYEE tuple with Ssn = '999887777' to '987654321'.

Result: Unacceptable, because it violates primary key constraint by repeating a value that already exists as a primary key in another tuple; it violates referential integrity constraints because there are other relations that refer to the existing value of Ssn.