# 1-Key Constraints

- 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)
  - Not always applicable – choice is sometimes subjective

**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 |

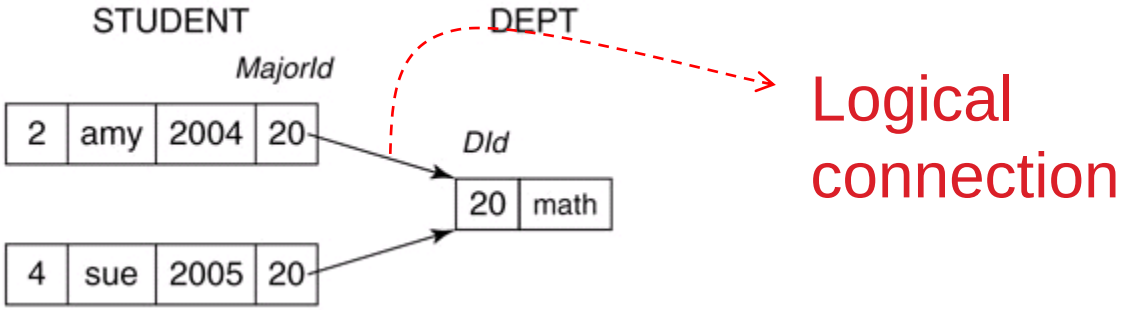
**Figure 5.4**
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

# 2-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.
    - t[PK] $\neq$ null for any tuple t in r(R)
    - If PK has several attributes, null is not allowed in any of these attributes
  - Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key. (*UNIQUE kısıtlaması*)

# 3-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.**
- 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.
- 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.*



# Other Types of Constraints

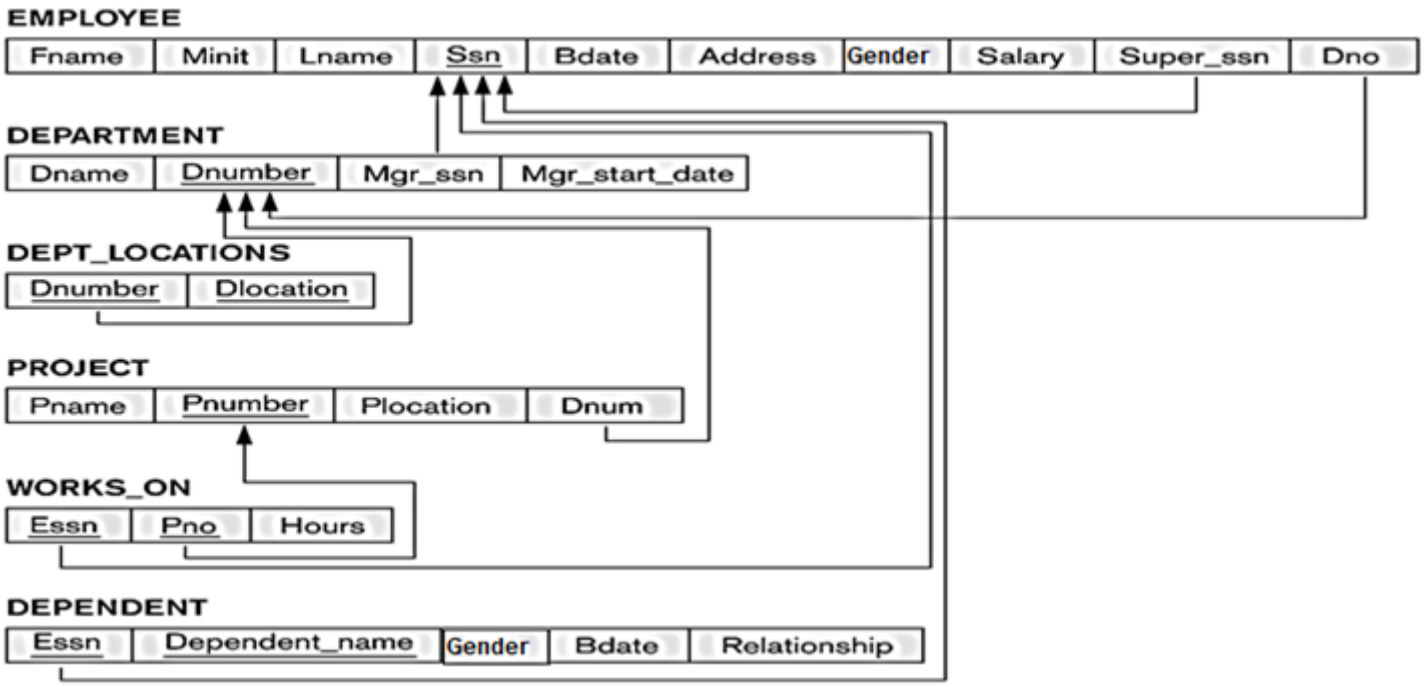
- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the model per se
  - Example: “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- A **constraint specification** language may have to be used to express these. SQL-99 allows triggers and **ASSERTIONS** to express for some of these

# Actions in case of integrity violation

- 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
- Specifying constraints **in SQL**
  - NOT NULL may be specified on an attribute
  - Key attributes can be specified via the PRIMARY KEY and UNIQUE phrases
  - referential integrity constraints (foreign keys).
  -

# 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
- COMPANY **relational schema diagram:**



# 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 (CRUD– Create, read, update and delete 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
- Next slide shows an example state for the COMPANY database

# Populated database state for COMPANY

**Figure 5.6**
One possible database state for the COMPANY relational database schema.

**EMPLOYEE**

| Fname | Minit | Lname | Ssn | Bdate | Address | Gen | 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 | Gen | 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 |

- 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.

# Specifying constraints in SQL

```
CREATE TABLE DEPT (
  DNAME    VARCHAR(10)  NOT NULL,
  DNUMBER INTEGER     NOT NULL,
  MGRSSN  CHAR(9) DEFAULT '000',
  MGRSTARTDATE       CHAR(9),
  PRIMARY KEY (DNUMBER),
  UNIQUE (DNAME),
  FOREIGN KEY (MGRSSN) REFERENCES EMP
                ON DELETE SET DEFAULT ON UPDATE CASCADE);
 CREATE TABLE EMP(
   ENAME     VARCHAR(30)NOT NULL,
   ESSN      CHAR(9),
   BDATE     DATE,
   DNO       INTEGER  DEFAULT 1,
   SUPERSSN CHAR(9),
   PRIMARY KEY (ESSN),
   FOREIGN KEY (DNO) REFERENCES   DEPT ON DELETE SET
   DEFAULT ON UPDATE  CASCADE,
   FOREIGN KEY (SUPERSSN) REFERENCES EMP ON DELETE SET
                NULL ON UPDATE CASCADE);
```

ÖNEMLİ: Yukarıdaki SQL DDL tablo oluşturma komutları 2 farklı sıra için de (DEPT, EMP ve EMP,DEPT) çalıştırmak mümkün olmuyor. Çünkü imalar mevcut omayan başka bir tabloya işaret edemez. O yüzden, ima kısıtlarını temel tabloyu oluşturduktan sonra ALTER TABLE ... yardımcı komutu ile ekleyebilriz.

Aynı durum DROP TABLE için de geçerli.

# Populate DB with SQL

- Aynı DİKKAT, INSERT TABLE için de geçerli. Bu sefer UPDATE komutu yardımı ile eksikler tamamlanır. O yüzden önceki sayfadaki VT'da eklemeler aşağıdaki gibi olması gerek: örneğin:

```
INSERT INTO EMP VALUES
  ('James', '888665555','10-NOV-27',null,null);
INSERT INTO EMP VALUES
  ('Franklin','T','Wong','333445555','08-DEC-45','638 Voss, Houston,
    TX','M',40000,'888665555',null);
.....
INSERT INTO DEPT VALUES ('Research', 5, '333445555', '22-MAY-78');
INSERT INTO DEPT VALUES ('Headquarters', 1, '888665555', '19-JUN-71');

UPDATE employee SET dno = 5 WHERE ssn = '333445555';
UPDATE employee SET dno = 1 WHERE ssn = '888665555';
```

# Exercise-1

(Taken from Exercise 5.15)

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course: **Draw a relational schema diagram**

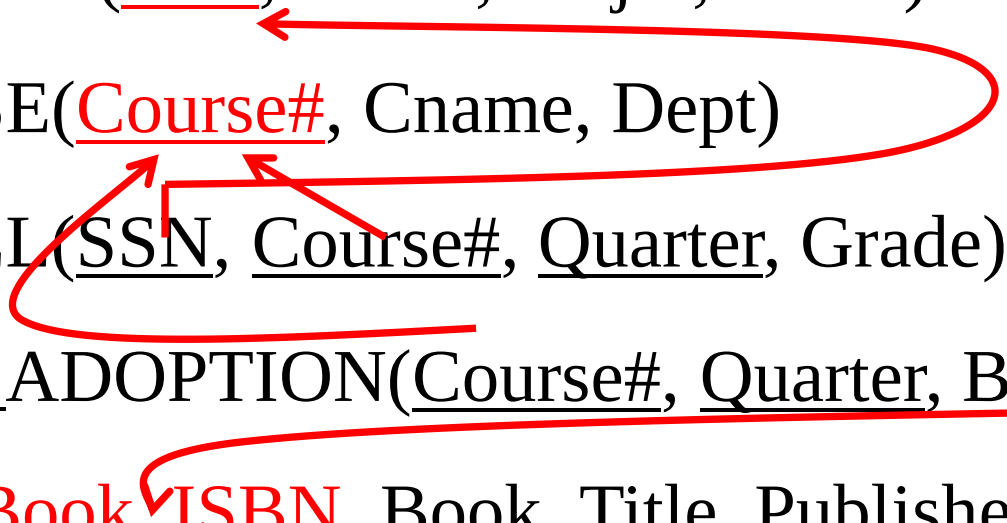
STUDENT(SSN, Name, Major, Bdate)

COURSE(Course#, Cname, Dept)

ENROLL(SSN, Course#, Quarter, Grade)

BOOK_ADOPTION(Course#, Quarter, Book_ISBN)

TEXT(Book_ISBN, Book_Title, Publisher, Author)



# Exercise-2 : possible violation when inserting

- 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

# Exercise-3: possible violation when deleting

- 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 (see Chapter 8 for more details)
      - RESTRICT option: reject the deletion
      - CASCADE option: remove all 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

# Exercise-4: possible violation when updating

- 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