Lecture 5: Integrity Constraints

CS3402 Database Systems

Integrity Constraints

- Constraints determine which values are permissible and which are not in the database (table)
  - Constraints are conditions that must hold on all valid relation states
- A relational database schema S is a set of relation schemes $S = \{R_1, R_2, \ldots, R_n\}$ and a set of integrity constraints IC
- Valid state vs. invalid state
  - Valid state: a state that satisfies all the constraints in the defined set of integrity constraints
  - Invalid state: A database state that does not obey all the integrity constraints

Three Main Types of Relational Integrity Constraints

- Inherent or Implicit Constraints: characteristics of relations, e.g., no duplicate tuples
- Schema-based or Explicit Constraints: Expressed in schemas by DDL (i.e., SQL)
- Application-based or Semantic constraints: These are beyond the expressive power of the model (i.e., cannot be expressed in the schemas of the data model) and must be specified and enforced by the application programs

Schema-based Constraints

- There are three main types of 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)

Keys of Relations

- A set of one or more attributes $\{A_1, A_2, \ldots, A_n\}$ is a key for a relation if:
  - The attributes functionally determine all other attributes of the relation
  - Relations are sets. It is impossible for two distinct tuples of R to agree on all $A_1, A_2, \ldots, A_n$
  - No proper subset of $\{A_1, A_2, \ldots, A_n\}$ functionally determines all other attributes of R, i.e., a key must be minimal
- A functional dependency (FD) on a relation R is a statement of the form: if two tuples of R agree on attributes $\{A_1, A_2, \ldots, A_n\}$ (i.e., the tuples have the same values in their respective components for each of these attributes), then they must also agree on another attribute, “B”, $\{A_1, A_2, \ldots, A_n\} \rightarrow B$

Keys of Relations: Example

- Movies(title, year, length, type, studioName, starName): title, year, starName $\rightarrow$ length, type, studioName
  - Attributes {title, year, starName} form a key for the relation Movie
  - Suppose two tuples agree on these three attributes: title, year, starName
  - They must agree on the other attributes, length, type and studioName
  - No proper subset of {title, year, starName} functionally determines all other attributes
  - {title, year} does not determine starName since many movies have more than one star
  - {year, starName} is not a key because we could have a star in two movies in the same year

Key Constraints (1/3)

- Superkey of R
  - A set of attributes that contains a key is called a superkey
  - It is a set of attributes superkey SK, e.g., $\{A_1, A_2\}$ of R with the following conditions:
    - No two tuples in any valid relation state r(R) will have the same value for SK
    - For any distinct tuples t1 and t2 in r(R), t1[SK] $\neq$ t2[SK] (i.e., different SK)
- Key of R
  - A “minimal” superkey
  - A key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

Key Constraints (2/3)

- Example: Consider the CAR relation schema
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - CAR has two keys:
    - Key1 = {State, Reg#}
    - Key2 = {SerialNo}
  - Both are also superkeys of CAR
  - {SerialNo, Make} is a superkey but not a key
- In general:
  - Any key is a superkey (but not vice versa)
  - Any set of attributes that includes a key is a superkey
  - A minimal superkey is also a key

Key Constraints (3/3)

- 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
- General rule: Choose as primary key the smallest of the candidate keys (in terms of size)

Entity Integrity

- The primary key attributes PK of each relation schema R cannot have null values in any tuple of R
  - Primary key values are used to identify the individual tuples
  - $t[PK] \neq$ null for any tuple t in 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

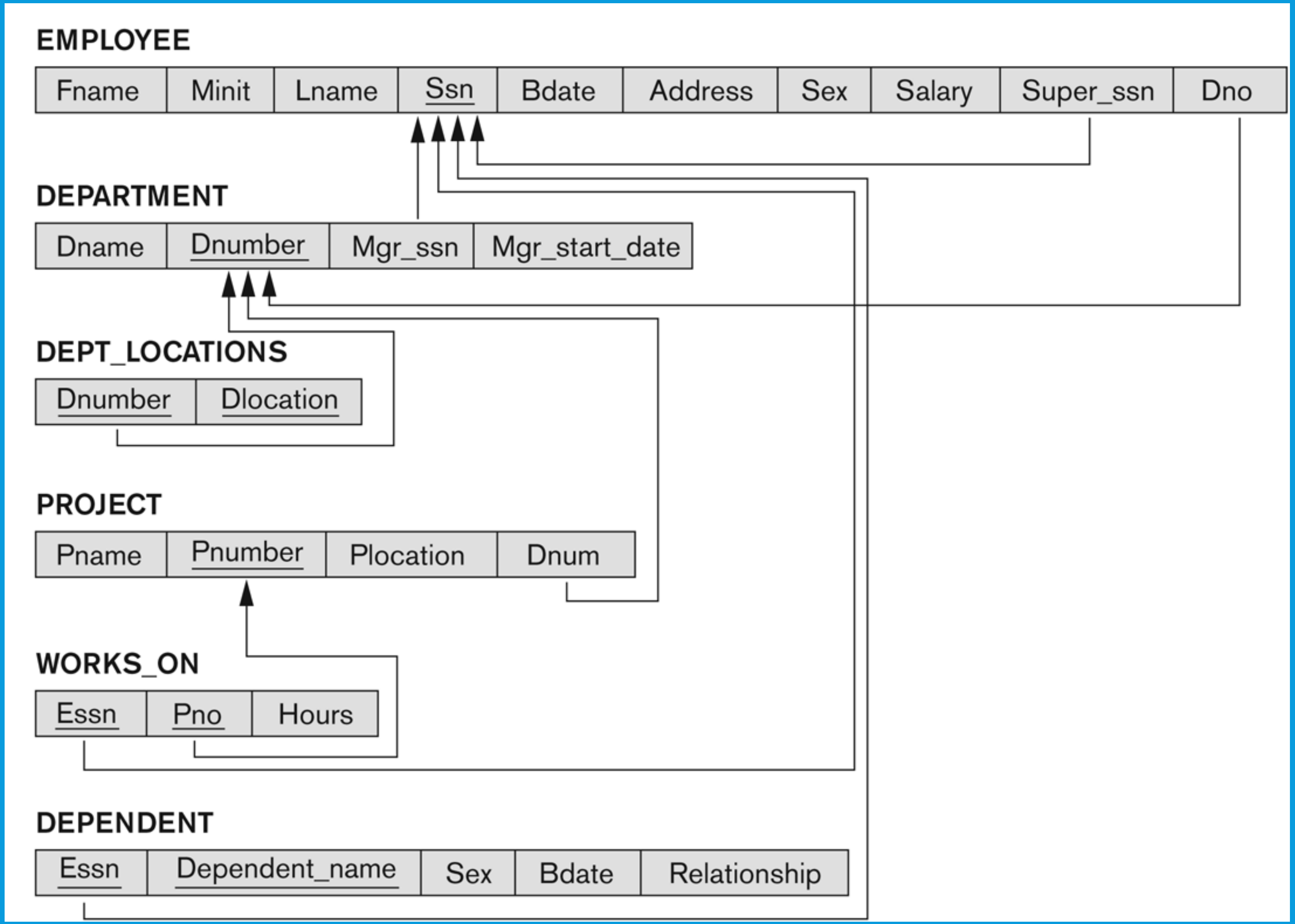
Referential Integrity

- Key and entity integrity constraints are specified on individual relations
- Referential integrity is a constraint involving two relations
  - To specify a relationship among tuples in two relations
  - The referencing relation and the referenced relation ($R_1 \rightarrow R_2$)
- 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 if it satisfies:
  - The attributes in FK have the same domain(s) as the primary key attributes PK of R_2

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
- Next slide shows the COMPANY relational schema diagram with referential integrity constraints

Referential Integrity Constraints for the COMPANY Relational Database Schema



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

SQL CREATE TABLE Data Definition Statements for COMPANY

```
CREATE TABLE EMPLOYEE
        ( FNAME             VARCHAR(15)       NOT NULL ,
          MINIT             CHAR ,
          LNAME             VARCHAR(15)       NOT NULL ,
          SSN               CHAR(9)           NOT NULL ,
          BDATE             DATE
          ADDRESS           VARCHAR(30) ,
          SEX               CHAR ,
          SALARY            DECIMAL(10,2) ,
          SUPERSSN          CHAR(9) ,
          DNO               INT               NOT NULL ,
   PRIMARY KEY (SSN) ,
   FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN) ,
   FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNUMBER) ) ;
CREATE TABLE DEPARTMENT
        ( DNAME             VARCHAR(15)       NOT NULL ,
          DNUMBER           INT               NOT NULL ,
          MGRSSN            CHAR(9)           NOT NULL ,
          MGRSTARTDATE      DATE ,
      PRIMARY KEY (DNUMBER) ,
      UNIQUE (DNAME) ,
      FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN) ) ;
CREATE TABLE DEPT_LOCATIONS
        ( DNUMBER           INT               NOT NULL ,
          DLOCATION         VARCHAR(15)       NOT NULL ,
      PRIMARY KEY (DNUMBER, DLOCATION) ,
      FOREIGN KEY (DNUMBER) REFERENCES DEPARTMENT(DNUMBER) ) ;
```

```
CREATE TABLE PROJECT
        ( PNAME             VARCHAR(15)       NOT NULL ,
          PNUMBER           INT               NOT NULL ,
          PLOCATION         VARCHAR(15) ,
          DNUM              INT               NOT NULL ,
      PRIMARY KEY (PNUMBER) ,
      UNIQUE (PNAME) ,
      FOREIGN KEY (DNUM) REFERENCES DEPARTMENT(DNUMBER) ) ;
CREATE TABLE WORKS_ON
        ( ESSN              CHAR(9)           NOT NULL ,
          PNO               INT               NOT NULL ,
          HOURS             DECIMAL(3,1)      NOT NULL ,
      PRIMARY KEY (ESSN, PNO) ,
      FOREIGN KEY (ESSN) REFERENCES EMPLOYEE(SSN) ,
      FOREIGN KEY (PNO) REFERENCES PROJECT(PNUMBER) ) ;
CREATE TABLE DEPENDENT
        ( ESSN              CHAR(9)           NOT NULL ,
          DEPENDENT_NAME    VARCHAR(15)       NOT NULL ,
          SEX               CHAR ,
          BDATE             DATE ,
          RELATIONSHIP      VARCHAR(8) ,
             PRIMARY KEY (ESSN, DEPENDENT_NAME) ,
             FOREIGN KEY (ESSN) REFERENCES EMPLOYEE(SSN) ) ;
```

Referential Integrity

- Referential integrity constraints typically arise from the relationships among the entities represented by the relation
- For example, in the EMPLOYEE relation, the attribute Dno refers to DEPARTMENT for which an employee works. We designate Dno to be a foreign key of EMPLOYEE referencing the DEPARTMENT.
- A value of Dno in any tuple t_1 of the EMPLOYEE relation must match a value of the primary key of DEPARTMENT, Dnumber, in the same tuple t_2 of the DEPARTMENT relation or the value of Dno can be NULL if the employee does not belong to a department or will be assigned to a department later.

Update Operations on Relations

- Update operations
  - INSERT a tuple
  - DELETE a tuple
  - MODIFY a tuple
- Integrity constraints should not be violated by the update operations
- Update the department number of “Research” from “001” to “R001”
- 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

Possible Violations for Update Operations

- DELETE may violate only referential integrity
  - If the primary key value of the tuple being deleted is referenced from other tuples in the database
- 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

Integrity Violation

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation
  - 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

Adding Constraints in SQL

```
CREATE TABLE TOY
(   toy_id              NUMBER(10),
    description         VARCHAR(15) NOT NULL,
    purchase_date       DATE,
    remaining_qnt       NUMBER(6));

CREATE TABLE TAB1
(   col1 NUMBER(10)  PRIMARY KEY,
    col2 NUMBER(4)   NOT NULL,
    col3 VARCHAR(5)  REFERENCES zipcode(zip) ON DELETE CASCADE,
    col4 DATE,
    col5 VARCHAR(20) UNIQUE,
    col6 NUMBER(5)    CHECK (col6 < 100));
```

A unique constraint is a single field or combination of fields that uniquely defines a record.

A check constraint allows you to specify a condition on each row in a table.

Naming Constraints in SQL

```
CREATE TABLE TAB2
(   col1 NUMBER(10),
    col2 NUMBER(4)    NOT NULL,
    col3 VARCHAR(5),
    col4 DATE,
    col5 VARCHAR(20),
    col6 NUMBER(5),
    CONSTRAINT TAB1_PK PRIMARY KEY(col1),
    CONSTRAINT TAB1_ZIPCODE_FK FOREIGN KEY(col3)
        REFERENCES ZIPCODE(zip) ON DELETE CASCADE,
    CONSTRAINT TAB1_COL5_UK UNIQUE(col5),
    CONSTRAINT TAB1_COL6_CK CHECK(col6 < 100));
```

A foreign key with cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted. This is called a cascade delete in Oracle.

Reference Constraints in SQL (1/2)

```
CREATE TABLE COUNTRY
(cntry_cd       VARCHAR(3)                    NOT NULL,
 cname          VARCHAR2(32)                  NOT NULL,
 ename          VARCHAR2(32)                  NOT NULL,
 curr_cd        VARCHAR(3)                    NOT NULL,
 upd_dt         DATE DEFAULT SYSDATE          NOT NULL ,
 upd_uid        VARCHAR2(16)                  NOT NULL);

CREATE TABLE EXCHANGE
(exchg_cd       VARCHAR2(8)                   NOT NULL,
 cname          VARCHAR2(32)                  NOT NULL,
 ename          VARCHAR2(32)                  NOT NULL,
 cntry_cd       CHAR(3)                       NOT NULL);
```

Reference Constraints in SQL (2/2)

- Add constraints to tables COUNTRY and EXCHANGE
  - ALTER TABLE COUNTRY ADD CONSTRAINT PK_country PRIMARY KEY(cntry_cd);
  - ALTER TABLE EXCHANGE ADD CONSTRAINT PK_exchange PRIMARY KEY(exchg_cd);
  - ALTER TABLE EXCHANGE ADD CONSTRAINT FK_exchg_cntry FOREIGN KEY(cntry_cd) REFERENCES COUNTRY(cntry_cd);