CS2102 Database Systems

THE RELATIONAL MODEL

Relational Model Basics

- The Relational Model was introduced by Edgar Codd from IBM Research Laboratory
- Data is modeled in terms of relations, tables with rows and columns
- Degree/Arity = No. of columns — no. of attributes
- Cardinality = No. of rows — no. of records

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Relational Model Basics

- A relational database is a set of relations — tables
- Each relation has a definition called a relation schema — logical schema = relation schema
  - Schema specifies name of relation, attributes and data constraints. — columns of a table
  - Data constraints include domain constraints

  Students (sid: **string**, sname: **string**, login: **string**, age: **integer**, cap: **real**)
- Each row in a relation is called a tuple/record (all rows are distinct)

Relational Model Basics

- A domain is a set of atomic values
  - Special value null is a member of each domain
  - A null value means value is not applicable or unknown
- A relation is a set of tuples — rows of a table
  - Consider a relation schema $R(A_1, A_2, \ldots, A_n)$ with n attributes
  - Let D_i be the domain of attribute A_i (a set of possible values for A_i)
  - Each instance of R is a subset of $\{(a_1, a_2, \ldots, a_n) \mid a_i \in D_i\}$
- A relational database schema is a set of relation schemas

note: for each attribute of a relation, there is a set of permitted values, called the domain of that attribute. Thus, the domain of the salary attribute of the instructor relation is the set of all possible salary values

A domain is considered atomic if elements of that domain are indivisible units (no internal fields within each attribute)

relation schema: consist of a list of attributes and their corresponding domains
example:
department(deptName:string, building:string, budget:float)

Integrity Constraints (ICs)

- Integrity constraint is a condition that must be true for any instance of the database;
  - ICs are specified when schema is defined.
  - ICs are checked when relations are updated.
- A legal instance of a relation is one that satisfies all specified ICs.
- DBMS enforces ICs - allows only legal instances to be stored.

different types of ICs:
1) domain constraints
2) key constraints
3) foreign key contraints

Integrity Constraints (ICs)

- ICs are based on the semantics of the real-world enterprise that is being described in the database relations.
- We can check a database instance to see if an IC is violated, but we can NEVER infer that an IC is true by looking at an instance.
  - An IC is a statement about *all possible* instances!
- Common types of ICs
  - Domain constraints - restricts attribute values of relations
  - Key constraints, foreign key constraints

ICs: Key Constraints

- A superkey is a subset of attributes in a relation that uniquely identifies its tuples
- No two distinct tuples in a relation have same values in all attributes of superkey
- What are the possible superkeys for the following relation?

sid, cid

sid	*cid*	*grade*
1	CS204	C
1	CS101	A
2	CS204	C
3	CS101	B
3	MA112	B

ICs: Key Constraints

- A key is a superkey that satisfies the additional property: No proper subset of the key is a superkey
- That is, a key is a minimal subset of attributes in a relation that uniquely identifies its tuples
- Key attribute values cannot be null
- A relation could have multiple keys called candidate keys
- One of the candidate keys is chosen as the primary key

key = candidate key

Although the attributes ID and name together can distinguish instructor tuples, their combination, **{**ID, name**}**, does not form a candidate key, since the attribute ID alone is a candidate key.

ICs: Foreign Keys Constraints

- A subset of attributes in a relation is a foreign key if it refers to the primary key of a second relation

or can be a null value

Enrols

sid	*cid*	*grade*
1	CS204	C
1	CS101	A
2	CS204	C
3	CS101	B
3	MA112	B

Referencing relation

Courses

cid	*cname*	*credits*
CS101	Programming Methodology	5
MA112	Discrete Mathematics	4
IT112	Java to C	2
CS204	Analysis of Algorithms	4

Referenced relation

- *cid* is a foreign key in Enrols and a primary key in Courses

ICs: Foreign Keys Constraints

- Foreign key constraint (referential integrity constraint): Each foreign key value in referencing relation must either
  - Appear as primary key value in the referenced relation, or
  - Be a null value

other table

- Can occur within same relation
- Constraints on Course_Prereq
  - cid is the primary key
  - Prereq is the foreign key

Course_Prereq

cid	*prereq*
CS101	null
CS204	CS101
CS332	CS101
MA101	null
MA211	MA101

Primary and Foreign Keys Constraints

- Which of the following updates violates the primary/foreign key constraints?
  - Insert tuple (CS101, "Java Programming", 4) into Courses — violates primary key
  - Delete tuple (MA112, "Discrete Mathematics", 4) from Courses — violates foreign key
  - Delete tuple (IT112, "Java to C", 2) from Courses — safe
  - Modify tuple (3, MA112, B) to (3, MA113, B) in Enrols — violates foreign key
  - Insert tuple (1, null, C) into Enrols — safe

Enrols

sid	*cid*	*grade*
1	CS204	C
1	CS101	A
2	CS204	C
3	CS101	B
3	MA112	B

Courses

cid	*cname*	*credits*
CS101	Programming Methodology	5
MA112	Discrete Mathematics	4
IT112	Java to C	2
CS204	Analysis of Algorithms	4

How to Handle Foreign Key Violation?

- Which can be done if a tuple *t* in Courses is to be deleted? or modified
- Options:
  - Disallow deletion if some row in Enrols refers to *t*
  - Delete all rows in Enrols that refer to *t*
  - For each row in Enrols that refers to *t*, replace its *cid* value with an existing default value
  - For each row in Enrols that refers to *t*, replace its *cid* value with the null value, provided *cid* is not a primary key attribute in Enrols
- Above options also apply if the primary key of a tuple in Courses is to be modified

Relational Query Languages

- A major strength of the relational model: it supports simple, powerful *querying* of data.
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.

Structured Query Language (SQL)

- Most widely used commercial relational database language
- Developed by IBM San Jose Research Laboratory in the 1970s
- Standards:
  - SQL-86
  - SQL-89 (minor revision)
  - SQL-92 (major revision)
  - SQL-99 (major extensions, current standard)

SQL: Table Definition

- CREATE TABLE statement
  - Specify attributes and constraints of table
- Basic attribute types
  - CHAR(n), VARCHAR(n), BOOLEAN, INTEGER, REAL, DATE, TIME
- Basic constraints
  - Primary, candidate, foreign key constraints

SQL: Table Definition

```
CREATE TABLE Courses (
    cid         CHAR(5)     DEFAULT `C0000',
    cname       VARCHAR(50),
    credits     INTEGER,
    prereq      CHAR(5),
    PRIMARY KEY (cid),
    UNIQUE (cname),
    FOREIGN KEY (prereq) REFERENCES Courses(cid)
)
```

- Type of each field is specified, and enforced by the DBMS when tuples are added or modified.
- Candidate keys are *cid* and *cname* (specified using UNIQUE)
- Each course *cid* has at most one prerequisite course

by default, primary key has to be unique and not null

SQL: Table Definition

```
CREATE TABLE Students (
    sid         CHAR(5),
    sname       VARCHAR(20),
    age         INTEGER,
    PRIMARY KEY (sid)
)

CREATE TABLE Enrols (
    sid  CHAR(5), cid  CHAR(5), grade  CHAR,
    PRIMARY KEY (sid, cid),
    FOREIGN KEY (sid)  REFERENCES  Students(sid),
    FOREIGN KEY (cid) REFERENCES Courses(cid)
)
```


A primary key can also be a foreign key (also occurs in 1-1 relationship)

Enforcing Referential Integrity

- *sid* in Enrols is a foreign key that references Students.
- What should be done if an Enrols tuple with a non-existent student id is inserted? (*Reject it!*)
- What should be done if a Students tuple is deleted?
  - Also delete all Enrols tuples that refer to it
  - Disallow deletion of a Students tuple that is referred to
  - Set *sid* in Enrols tuples that refer to it to a *default sid*
  - Set *sid* in Enrols tuples that refer to it to *null*
- Similar if primary key of Students tuple is updated.

default

SQL: Options for Foreign Key Violations

```
CREATE TABLE Enrols (
    sid             CHAR(5),
    cid             CHAR(5),
    grade           REAL,
    PRIMARY KEY (sid, cid),
    FOREIGN KEY (sid) REFERENCES Students(sid),
    FOREIGN KEY (cid) REFERENCES Courses(cid)
        ON DELETE/UPDATE option
)
```

Options:

- **NO ACTION** reject any violations (default)
- **CASCADE** propagate delete/update to referencing tuples
- **SET DEFAULT** update foreign keys of referencing tuples to some default values
- **SET NULL** update foreign keys of referencing tuples to null values

SQL: Options for Foreign Key Violations

```
CREATE TABLE Students (
    sid             CHAR(5),
    sname           VARCHAR(20),
    age             INTEGER,
    PRIMARY KEY (sid),
)

CREATE TABLE Courses (
    cid             CHAR(5)   DEFAULT `C0000',
    cname           VARCHAR(50),
    credits         INTEGER,
    PRIMARY KEY (cid),
)

CREATE TABLE Enrols (
    sid CHAR(5), cid CHAR(5), grade REAL,
    PRIMARY KEY (sid, cid),
    FOREIGN KEY (sid) REFERENCES Students(sid)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
    FOREIGN KEY (cid) REFERENCES Courses(cid)
        ON UPDATE CASCADE
        ON DELETE SET DEFAULT
)
```

what happens if a tuple with the default value is deleted?

SQL: Query Single Relation

- Find all 18 year old students

```
SELECT *
FROM   Students S
WHERE  S.age = 18
```

Students

sid	name	login	age	cap
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Find student names and logins

```
SELECT S.name, S.login
FROM   Students S
```


SQL: Query Multiple Relations

Students

sid	name	login	age	cap
53831	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enrols

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53688	History105	B

```
SELECT S.name, E.cid
FROM   Students S, Enrols E
WHERE  S.sid = E.sid AND E.grade = "A"
```

S.name	E.cid
Smith	Topology112

SQL: Destroy Table

```
DROP TABLE  Students
```

- Destroys the relation Students.
- The schema information and the tuples are deleted.

SQL: Alter Table

```
ALTER TABLE  Students
        ADD COLUMN firstYear: integer
```

- Schema of Students is altered by adding a new attribute
- Every tuple in the current instance of the relation is extended with a *null* value in the new field.

SQL: Adding and Deleting Tuples

- Insert a single tuple

```
INSERT INTO  Students (sid, name, login, age, gpa)
VALUES  (53688, 'Smith', 'smith@ee', 18, 3.2)
```

- Delete all tuples satisfying some condition (e.g., name = Smith)

```
DELETE
FROM    Students S
WHERE  S.name  =  'Smith'
```

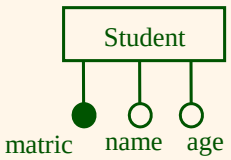

Summary

- Relational model is a tabular representation of data.
- Integrity constraints is based on application semantics.
  - Domain constraints
  - Primary/candidate key constraints
  - Foreign key constraints
- Options for dealing with foreign key constraint violations
  - Disallow violations
  - Update/delete referencing tuples

From ER to RELATIONAL

Map Entity Types

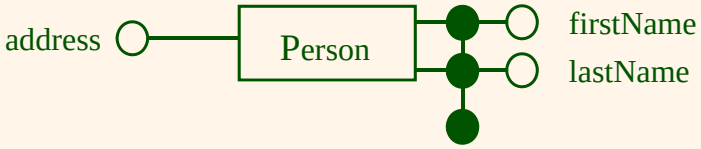
- Map an entity type to a relation
- Entity type attributes become attributes of the relation
- Key of entity type becomes key of relation



Students (matric, name, age)

```
CREATE TABLE Students (
        matric   VARCHAR(5),
        name     VARCHAR(20),
        age      INTEGER,
        PRIMARY KEY (matric)
)
```

Map Entity Types

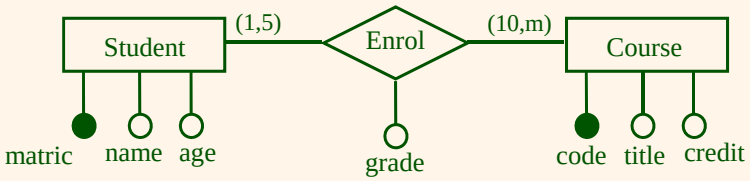



```
CREATE TABLE Person (
        firstName VARCHAR(32),
        lastName VARCHAR(32),
        address VARCHAR(128),
        PRIMARY KEY (firstName, lastName)
)
```


Map Relationship Sets

- Map a relationship set to a relation
- Attributes of relationship set become attributes of relation
- Identifier of each participating entity type form a superkey for the relation
  - They are also foreign keys

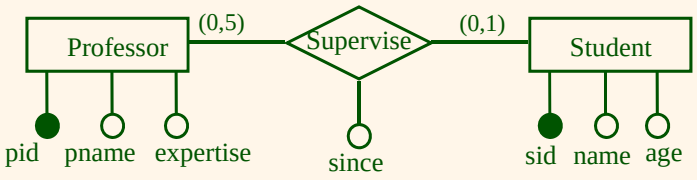
Many-to-Many Relationship



Enrol (matric, code, grade)

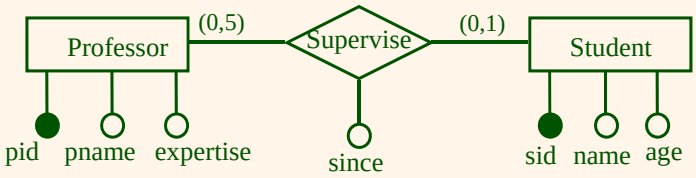
```
CREATE TABLE Enrol (
        matric  CHAR(5),  code  CHAR(5),  grade  CHAR,
        PRIMARY KEY (matric, code),
        FOREIGN KEY (matric)  REFERENCES  Student(matric),
        FOREIGN KEY (code)  REFERENCES Course(code)
)
```


Many-to-One Relationship



- Each student is supervised by at most one professor
- Map relationship type Supervise to a relation
- Key is sid
- Separate tables for Professor and Student.
- Since each student has a unique professor, we could also combine Student and Supervise.

Many-to-One Relationship



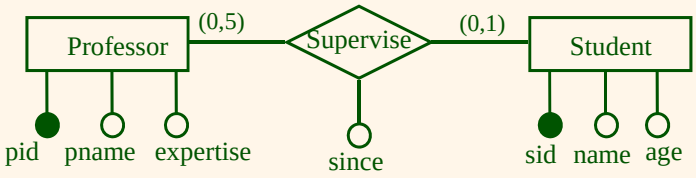
Professor (pid, pname, expertise)
Student (sid, name, age)
Supervise (sid, pid, since)

OR

pid not part of superkey in supervise table

Professor (pid, pname, expertise)
SupervisedStudent (sid, sname, age, pid, since)

Many-to-One Relationship



primary key does not need to be formed from pid, since sid alone is sufficient, as each student will only have 1 supervisor

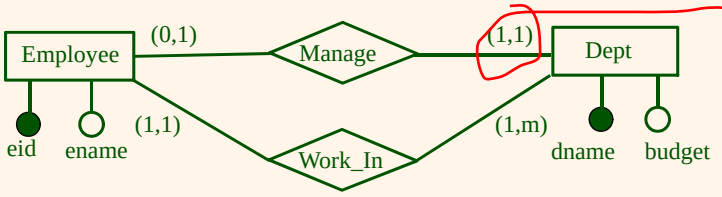
```
CREATE TABLE Supervise (
   sid   CHAR(5),
   pid   CHAR(8),
   since  DATE,
   PRIMARY KEY (sid),
   FOREIGN KEY (sid)
         REFERENCES  Student(sid),
   FOREIGN KEY (pid)
         REFERENCES Professor(pid)
)
```

OR

```
CREATE TABLE SupervisedStudent (
   sid   CHAR(5),  sname  CHAR(20),
   age  INTEGER,
   pid   CHAR(8),
   since  DATE,
   PRIMARY KEY (sid),
   FOREIGN KEY (pid)
         REFERENCES Professor(pid)
)
```

Map Participation Constraints

- Does every department have a manager?
- If so, the participation of Dept in Manage is *mandatory* or *total* (vs. *optional* or *partial*)
- Every *dname* value in Dept table must appear in a row of the Manage table (with a non-null *eid* value)



mandatory: every dept must appear in the manage table

Map Participation Constraints

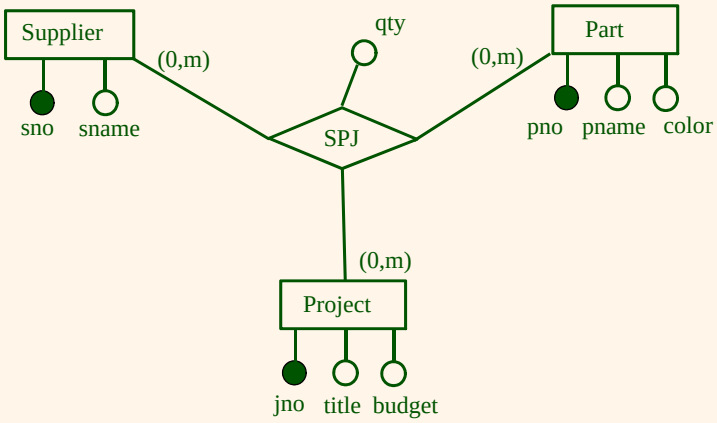
- Capture participation constraints involving one entity type in a binary relationship

```
CREATE TABLE  DeptManager (
  dname          CHAR(20),
  budget         REAL,
  eid            CHAR(5) NOT NULL,
  PRIMARY KEY  (dname),
  FOREIGN KEY  (eid) REFERENCES Employee,
  ON DELETE NO ACTION
)
```

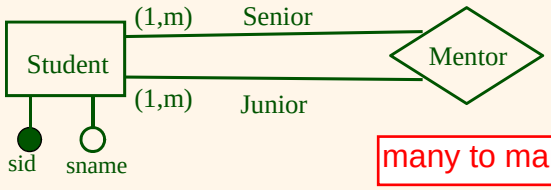
id of manager

Exercise

What are the relations obtained from this ER diagram?



Map Recursive Relationship



many to many

Mentor (senior_sid, junior_sid)

```
CREATE TABLE Mentor (
        senior_sid   CHAR(5),  junior_sid   CHAR(5),
        PRIMARY KEY (senior_sid, junior_sid),
        FOREIGN KEY (senior_sid)  REFERENCES Student(sid),
        FOREIGN KEY (junior_sid)  REFERENCES Student(sid)
)
```

both the senior_sid and junior_sid are foreign keys, they reference the sid in the student table

Map Weak Entity Types

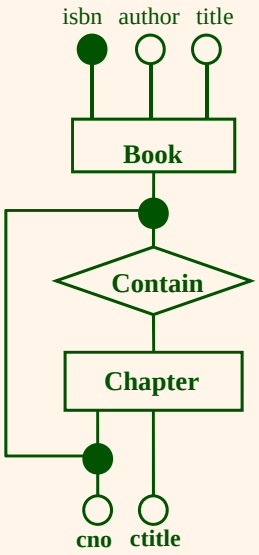
- Owner entity type and weak entity type must participate in a one-to-many relationship set (1 owner, many weak entities).
- Weak entity type must have total participation in this identifying relationship set.
- Weak entity type and identifying relationship set are mapped to a single relation
  - When the owner entity is deleted, all owned weak entities must also be deleted.

Map Weak Entity Types

Book (isbn, author, title)
Chapter (isbn, cno, ctitle)

```
CREATE TABLE Book (
   isbn   CHAR(10),   title   CHAR(50),
   author  CHAR(50),
   PRIMARY KEY (isbn) )

CREATE TABLE Chapter (
   isbn    CHAR(10),   cno  INTEGER, ctitle   CHAR(50),
   PRIMARY KEY (isbn, cno),
   FOREIGN KEY (isbn)  REFERENCES  Book(isbn),
   ON DELETE  CASCADE) )
```

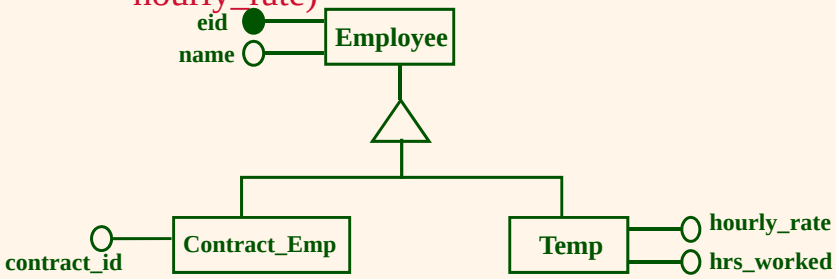


cascade: if a book were to be deleted, it has to be propagated to delete all chapters that belong to this book

Map ISA Hierarchies

- One relation per subclass or superclass

Employee (eid, ename)
Contract_Emp (eid, contractid)
Temp (eid, hrs_worked,
hourly_rate)



each entity from the sub classes have an eid, which is the primary and foreign key that references the employee table

Map ISA Hierarchies

- One relation per subclass or superclass

```
CREATE TABLE Employee (
   eid      CHAR(5)  PRIMARY KEY,
   ename  CHAR(30),
)

CREATE TABLE Contract_Emp (
   eid      CHAR(5)  PRIMARY KEY
            REFERENCES Employee,
   contractid  CHAR(10)
)

CREATE TABLE Temp (
   eid      CHAR(5)  PRIMARY KEY
            REFERENCES Employee,
   hrs_worked      INTEGER,
   hourly_rate      REAL
)
```

Map ISA Hierarchies

- One relation per subclass if each employee must be in one of these two subclasses.

Contract_Emp (eid, ename, contractid)
Temp (eid, ename, hrs_worked, hourly_rate)

```
CREATE TABLE ContractEmp (
   eid       CHAR(5)  PRIMARY KEY,
   ename CHAR(30),
   contractid  CHAR(10)
)
```

```
CREATE TABLE Temp (
   eid       CHAR(5)  PRIMARY KEY,
   ename CHAR(30),
   hrs_worked      INTEGER,
   hourly_rate      REAL
)
```


Summary

- Rules for mapping ER to relational model
  - Entity types and relationship sets
  - Relationship cardinalities
  - Participation constraints
  - Recursive relationship
  - Weak entity types
  - ISA hierarchies