Lecture 02: Relational Database Model

CS 1555: Database Management Systems

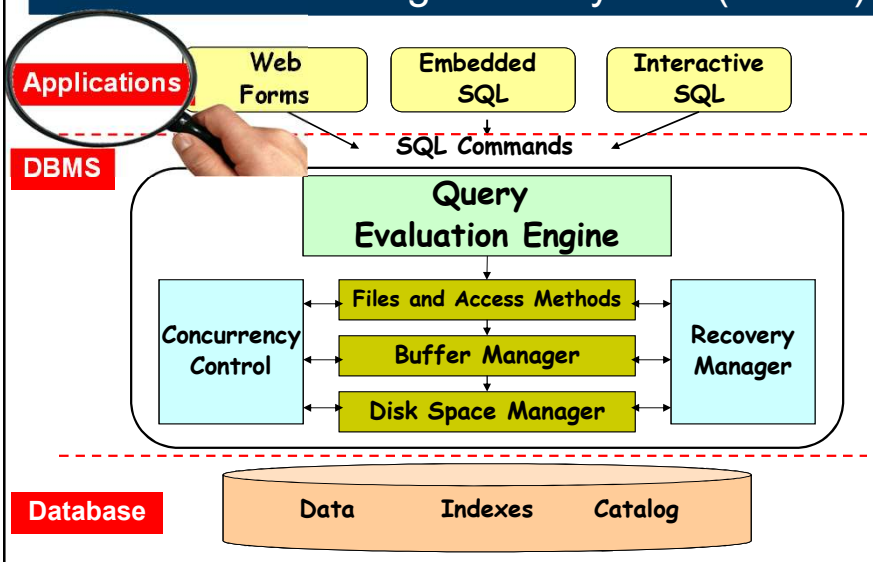
Constantinos Costa

www.cs.pitt.edu/~nlf4/cs1555/ (tentative)

Jan 10, 2019, 16:00-17:15
University of Pittsburgh, Pittsburgh, PA

Lectures based: P. Chrysanthis & N. Farnan Lectures

Database Management System (DBMS)

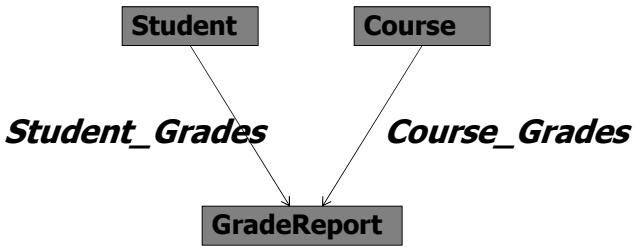


The relational data model

- Proposed by E.F. “Ted” Codd in 1970
  - “A Relational Model of Data for Large Shared Data Banks.”
  - Built on the concept of the mathematical relation
  - Codd won a Turing award for this work in 1981
- First systems came about in 1977-1978
  - System-R and Ingres
- First commercial systems in the 1980’s
  - IBM and Oracle

Example of CODASYL Database

Student Records



FIND ANY <record-type> USING <fields list>
GET {FIRST, NEXT, LAST} MEMBER WITHIN <set-type>
WHERE <condition>

Relational Model

SID	Name	Major	GPA
546007	Susan	CS	3.80
546100	Bob	CoE	3.65
546500	Bill	CS	3.70

Students

CID	Name
CS 1555	DB
CS 1530	SW
CS 1550	OS

Courses

SID	CID	Grade
546007	CS 1550	A
546007	CS 1530	B+
546100	CS 1550	B

Enrollment

- It is the most popular implementation model
  - Simplest, most uniform data structures, and is the most formal of all data model
- Both entity types and relationship types are represented by ***relations***, i.e., **tables**

The Mathematical Concept of Relation

- Let D_1, D_2,..., D_n be domains (not necessarily distinct), the Cartesian product of these *n* sets

  $$D_1 \times D_2 \times \ldots \times D_n$$

  is the set of all possible ordered n-tuples

  $(v_1, v_2, \ldots, v_n)$ such that $v_1 \in D_1$, $v_2 \in D_2$, ..., $v_n \in D_n$
- E.g., let D_1= {Nick, Susan} and D_2= {BS, MS, PhD}

  $D_1 \times D_2$ = {(Nick, BS), (Nick, MS), (Nick, PhD), (Susan, BS), (Susan, MS), (Susan, PhD)}
- A relation is any subset of the Cartesian product
  - R_1= {(Nick, BS),(Nick, MS), (Susan, BS), (Susan, PhD)}
  - R_2= {}

Two Notations

- Relation schema R is denoted by

  $R = \{A_1:D_1, A_2:D_2,..., A_n:D_n\}$ or $R = \{A_1, A_2,..., A_n\}$

  or $R (A_1, A_2,..., A_n)$

  – defines the name, the names of the attributes of that relation, and domains of those attributes

- Set-of-attributes

  – A tuple t of r(R) is denoted by

  $t = \{A_1:v_1, A_2:v_2,..., A_n:v_n\}$, $v_i \in D_i$, $1 \le i \le n$ or

  $t = <(A_1:v_1), (A_2:v_2),..., (A_n:v_n)>$, $v_i \in D_i$, $1 \le i \le n$

- List of attributes

  – A tuple t of r(R) is denoted by

  $t = (v_1, v_2,..., v_n)$, $v_i \in D_i$, $1 \le i \le n$

SQL Insert

STUDENT(SID, Name, Major, GPA)

- **Implicit (list)**:

```
INSERT INTO STUDENT
  VALUES (165, `Susan Jones`, 'CS', 0.00);
```

- **Explicit (set)**:

```
INSERT INTO STUDENT (SID, Name)
  VALUES (165, `Susan Jones`);

INSERT INTO STUDENT (Name, SID)
  VALUES ( `Susan Jones`, 165);
```

- Values-clause may be a list of tuples in some systems

Properties of Relations

- A relation is finite
- There are no duplicate tuples in a relation
  - Recall a relation is a set of tuples
- Order of tuples in a relation is not important
  - Many logical orders can be specified on a relation
- A value may appear multiple times in a column
- Order of attribute values in a tuple is
  - important in a *list-of-attributes* definition
  - not important in a *set-of-attributes* definition

Relation Schema

STUDENT

SID	*LName*	*Name*	*Class*	*Major*
123	Smith	John	3	CS
395	Aiken	Mary	4	CS

Schema

- ***What is the meaning*?**
- A relation schema R specifies
  - The name of the relation
  - the attribute names A_i of R
  - the domain D_i (data type + format) for each attribute A_i
- data type is a set of **atomic data** values:
  - no attribute is a set-valued (1st Normal Form, 1-NF)
  - no attribute is composite
- format specifies the representation of a data value

Example Table Schema

Schema of STUDENT(SID, Name, Major, GPA)

```
CREATE TABLE STUDENT
(    SID     INTEGER,
     Name  CHAR(20),
     Major   CHAR(4),
     GPA    DEC(3,2)
);
```

Creating a Schema

- **Corresponding database is at an empty state!**
- **Initial state when the database is populated (loaded)**
- **Domain (type) of each field is specified and enforced by the DBMS whenever tuples are added or modified**

Example: Domain Constraints

SID	Name	Login	Age	GPA
546007	Jones	jones@cs	18	3.4
546100	Smith	smith@ee	18	3.2
546500	Smith	smith@math	19	3.8

- Example of **IC Violation**:

```
UPDATE  Students S
SET S.Age = `Eighteen`
WHERE S.Name = Jones;
```

Useful Terms

- Cardinality of a relation r(R): # of tuples in r(R) (denoted by |r(R)|)
- Arity or degree of r(R): # of attributes in R (denoted by |R|)

|r(R)| = 3

|R| = 4

SID	Degree	Major	Year
123	BS	Math	1992
064	BA	History	1991
445	PhD	CS	1999

- **|r(R)| ≥ 0 And |R| > 0**
- **Cardinality is property of a relation**
- **Arity is property of relation schema or a relation**

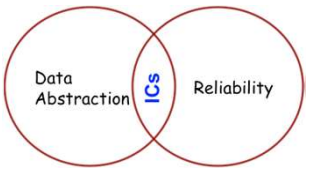
Relational Database Schema

- **A *database schema* is a set of relation schemas**
  and a set of *integrity constraints*

- Integrity Constraints
  - *Structural* Integrity Constraints
    - key constraints: uniqueness of keys
    - entity integrity constraint: no primary key value can be **NULL**
    - referential integrity constraint
  - *Semantic* Integrity Constraints
    - E.g., ??

Integrity Constraints (ICs)

- **IC**: condition that must be true for *any* instance of the database (e.g., domain constraints)
  - A **legal** instance of a relation is one that satisfies all specified ICs
  - ICs are specified when schema is **defined**
  - ICs are enforced when tables are **modified**



Example Table Schema in SQL (2)

Schema of STUDENT(SID, SSN, Name, Login, GPA)

```
CREATE TABLE STUDENT
(   SID     INTEGER  NOT NULL,
    SSN     CHAR(9),
    Name    CHAR(20),
    Login   CHAR(15),
    GPA     DEC(3,2),
    CONSTRAINT STUDENT_PK  PRIMARY KEY (SID),
    CONSTRAINT STUDENT_UN_SSN
        UNIQUE (SSN)
    CONSTRAINT STUDENT_UN_Login
        UNIQUE (Login)
);
```

Example Table Schema in SQL (3)

Schema of STUDENT(SID, SSN, Name, Login, GPA)

```
CREATE TABLE STUDENT
(   SID     INTEGER
        CONSTRAINT STUDENT_PK_NOT_NULL NOT NULL
        CONSTRAINT STUDENT_PK  PRIMARY KEY,
    SSN     CHAR(9)
        CONSTRAINT STUDENT_UN_SSN UNIQUE,
    Name    CHAR(20),
    Login   CHAR(15),
    GPA     DEC(3,2),
    CONSTRAINT STUDENT_UN_Login
        UNIQUE (Login)
);
```

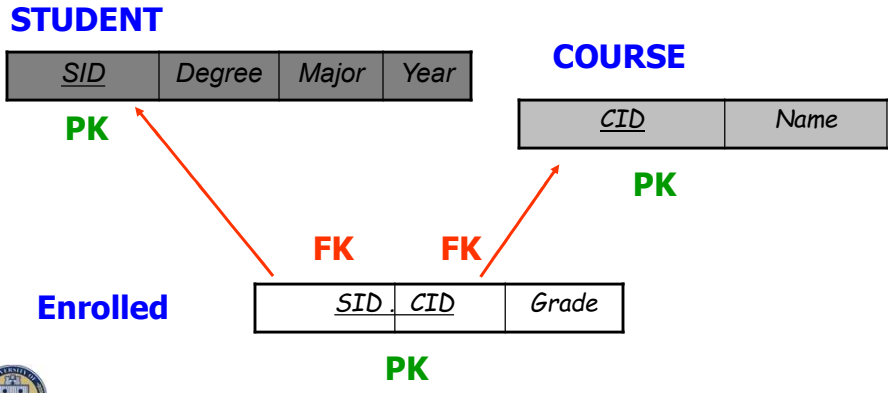
Identifying the Key

- What is the key in relation GRADUATE=(SID, Degree, Major, Year) ?

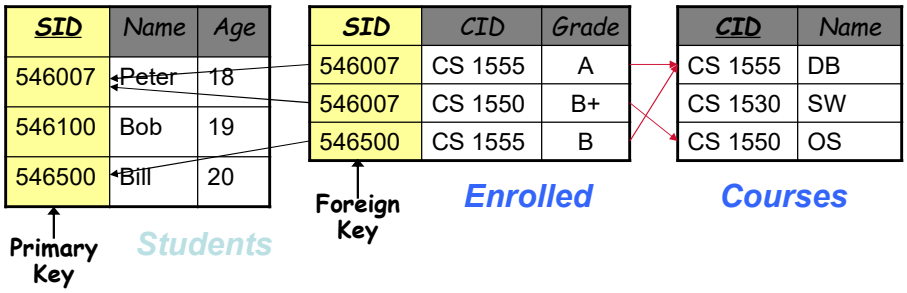
SID	Degree	Major	Year
123	BS	CS	1992
123	MS	CS	1993
064	BA	History	1991
445	PhD	CS	1999
123	BS	Math	1992
123	MS	Math	1992

Foreign Keys

- ***Foreign key*** **(FK) in relation R_2 is a set of attributes of R_2 that forms a primary key (PK) of another relation R_1.**
  - **Attributes in FK and PK have the same domain**



Foreign Key & Primary Key

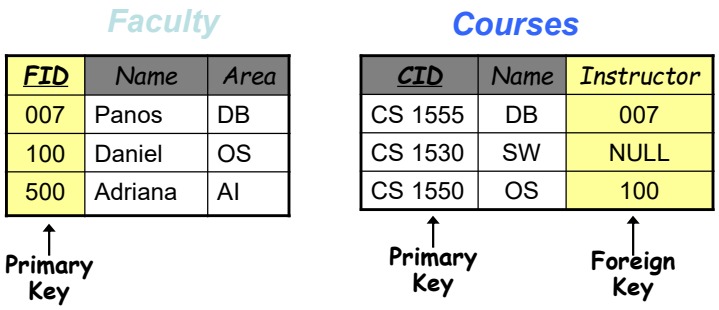


- **Foreign key:** Set of fields in one relation that is used to "*refer*" to a tuple in another relation
  - Must correspond to primary key of the referred relation
  - E.g. *SID* is a foreign key referring to *Students*

Foreign Key Constraints

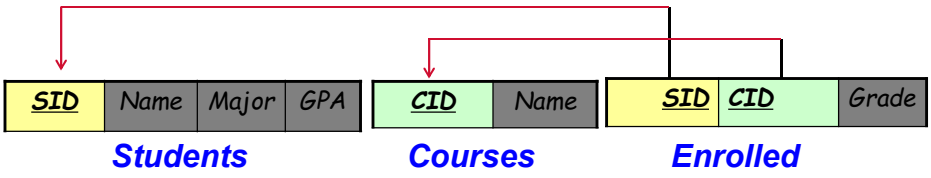
- If foreign key constraints are enforced, **referential integrity** is achieved
  - E.g.: Only students can enroll in a class
    - Only students listed in the "Students" relation should be allowed to enroll for courses
- Like a "*logical pointer*"
  - There shouldn't be dangling references
    - Either valid PK or NULL

Any Attribute can be a Foreign Key



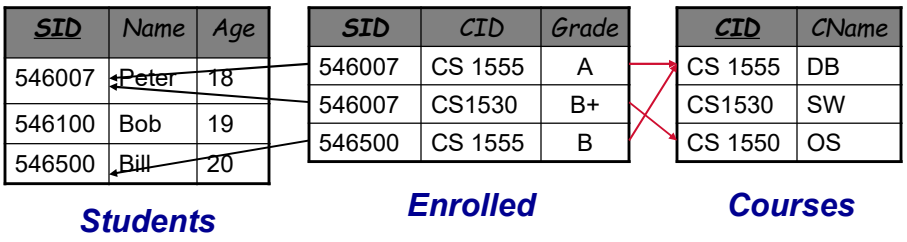
- **Foreign key:** Set of fields in one relation that is used to "*refer*" to a tuple in another relation
  - Must correspond to primary key of the referred relation
  - If not part of a key, it could be NULL

Foreign Keys in SQL




```
CREATE TABLE Enrolled (
  SID CHAR(20), CID CHAR(20), Grade CHAR(2),
  CONSTRAINT Enrolled_PK PRIMARY KEY (SID, CID)
,
  CONSTRAINT Enrolled_FK_sid
     FOREIGN KEY (SID) REFERENCES Students (SID)
,
  CONSTRAINT Enrolled_FK_cid
     FOREIGN KEY (CID) REFERENCES Courses
);
```

Referential Integrity Constraints



SID	Name	Age
546007	Peter	18
546100	Bob	19
546500	Bill	20

Students

SID	CID	Grade
546007	CS 1555	A
546007	CS1530	B+
546500	CS 1555	B

Enrolled

CID	CName
CS 1555	DB
CS1530	SW
CS 1550	OS

Courses

- Any IC Violation?

```
DELETE
FROM Enrolled E
WHERE E.SID = 546500;
```

No Violations! ✓

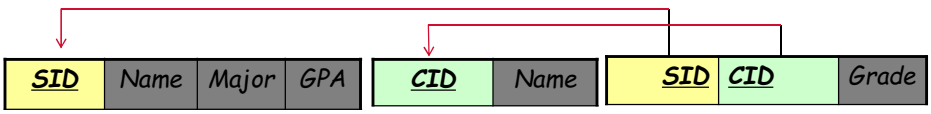
Referential Integrity Enforcement

- What are the alternatives when a "Students" tuple is **deleted**?
  1. **Delete all** Enrolled tuples that refer to it
  2. **Disallow** deletion of a Students tuple that is referred to
  3. **Set** *SID* in Enrolled tuples that refer to it to some "default" *SID* (e.g., 000000)
  4. **If** *SID was not part of the primary key,* **Set** *SID* to a special value "NULL", denoting *"unknown"* or *"inapplicable"*

Referential Integrity in SQL

- SQL/92 and SQL/99 support all 4 options on delete and update:
  - **`NO ACTION`** (default)
    - delete/update is rejected
  - **`CASCADE`**
    - also delete all tuples that refer to deleted tuple
  - **`SET NULL / SET DEFAULT`**
    - sets foreign key value of referencing tuple

RI Trigger Actions in SQL




```
CREATE TABLE Enrolled (
    SID CHAR(20), CID CHAR(20), Grade CHAR(2),
    CONSTRAINT Enrolled_PK PRIMARY KEY (SID, CID)
   ,
    CONSTRAINT Enrolled_FK_sid
        FOREIGN KEY (SID) REFERENCES Students (SID)
   ,
    CONSTRAINT Enrolled_FK_cid
        FOREIGN KEY (CID) REFERENCES Courses
        ON UPDATE CASCADE ON DELETE NO ACTION
);
```

Enforcing Integrity Constraints

- What would be the outcome?
  - Insert (585811, 'Jie', 19, 3.95) into Students
  - Insert (585811, NULL, NULL) into Enrollment
  - Insert (546100, 'CS 1555', NULL) into Enrollment
  - Insert (546100, 'Mary', 18, 3.65) into Students
  - Delete ('CS 1530') from Courses

SID	Name	Age	GPA
546007	Susan	18	3.8
546100	Bob	19	3.65
546500	Bill	20	3.7

Students

CID	Name
CS 1555	DB
CS 1530	SW
CS 1550	OS

Courses

SID	CID	Grade
546007	CS 1550	A
546007	CS 1530	B+
546100	CS 1550	B

Enrollment