The Relational Data Model

CSE 305 – Principles of Database Systems

Paul Fodor

Stony Brook University

http://www.cs.stonybrook.edu/~cse305

What Is a Data Model? Data and Its Structure

- All data is recorded as bits and bytes on a disk, but it is difficult to work with data at that level.
- It is convenient to view data at different *levels of abstraction*.
  - Programmers prefer to work with data stored in *files*.
    - Files belong to the physical level of data modeling.
    - File structures:
      - *Sequential files* are best for applications that access records in the order in which they are stored.
      - *Direct access* (or *random access*) files are best when records are accessed in unpredictable order. Files have *indices* (auxiliary data structures that enable applications to retrieve records based on the value of a *search key*).
- ***Schema***: Description of data at some abstraction level.
  - Each level has its own schema.
    - We will be concerned with three schemas: *physical*, *conceptual*, and *external*.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Physical Data Level

- *Physical schema* describes details of how data is stored: cylinders, tracks, indices etc.
- Early applications worked at this level – explicitly dealt with the details above (physical cylinders, tracks).
- **Problem:** Routines were hard-coded to deal with physical representation
  - Changes to data structure are difficult and very expensive to make.
  - Application code becomes complex since it must deal with details.
  - Rapid implementation of new features is impossible.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Conceptual Data Level

- The *Conceptual Data Level* hides the details of the physical data representation and instead describes data in terms of higher-level concepts that are closer to the way humans view it.
  - In the *relational model*, the conceptual schema presents data as a set of tables:

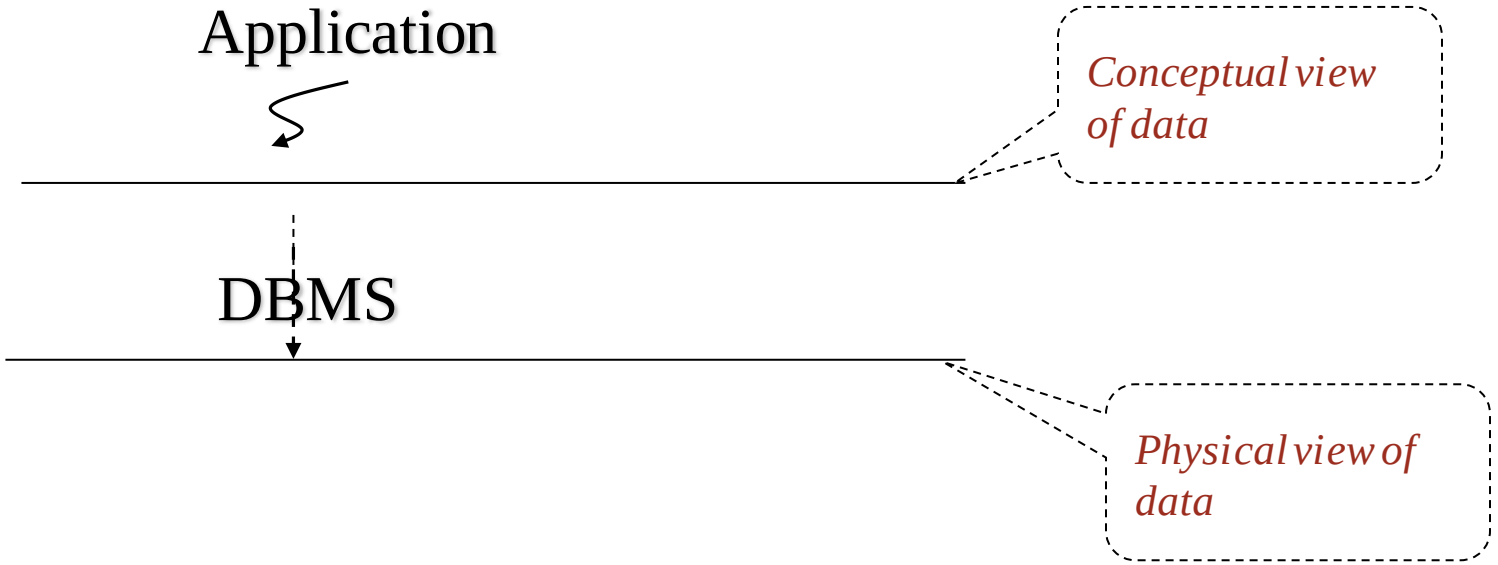
Student (Id: INT, Name: STRING, Address: STRING, Status: STRING)

- DBMS maps from conceptual to physical schema automatically.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Conceptual Data Level

- Physical schema can be changed without changing application:
  - DBMS would change mapping from conceptual to physical transparently
  - This property is referred to as ***physical data independence***



(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

External Data Level

- The *External Data Level* customize the conceptual schema to the needs of various classes of users (it also plays a role in database security).
  - In the relational model, the *external schema* also presents data as a set of relations.
  - An external schema specifies a *view* of the data in terms of the conceptual level.
- The external schema looks and feels like a conceptual schema, and both are defined in essentially the same way in modern DBMSs.
  - There might be several external schemas (i.e., views on the conceptual schema), usually one per user category.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

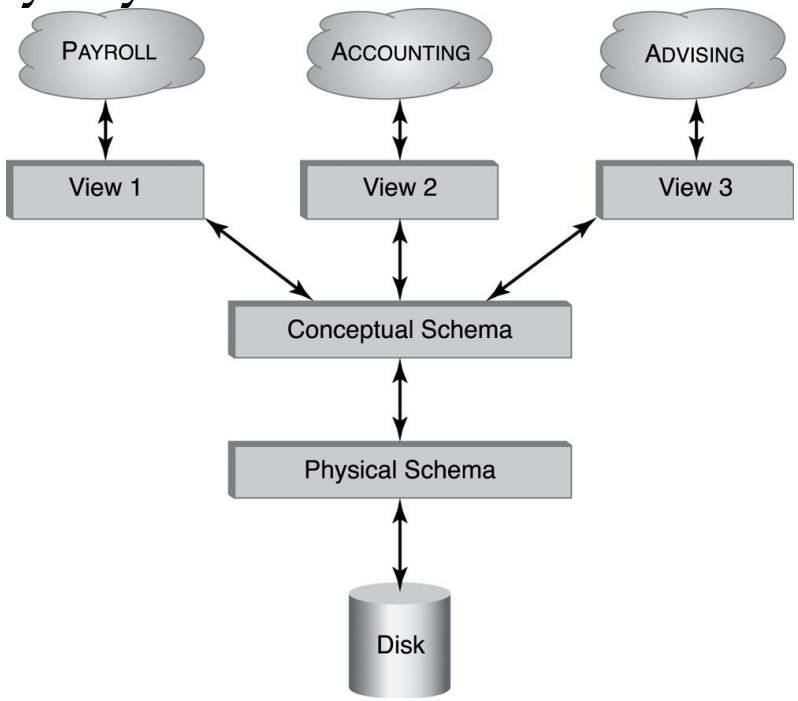
External Data Level

- It is tailored to the needs of a particular category of users.
  - Portions of stored data should not be seen by some users:
    - Students should not see their colleagues data (HWs, IDs, grades).
    - Faculty should not see billing data.
  - Information that can be derived from stored data might be viewed as if it were stored:
    - GPA not stored, but calculated when needed.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

External Data Level

- Applications are written in terms of external schemas.
- A view is computed when accessed (not stored).
- Translation from external to conceptual done automatically by DBMS at run time.



(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

External Data Level

- Conceptual schema can be changed without changing application (referred to as *conceptual data independence*):
  - Then, only the mapping from external to conceptual must be changed.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Data Model

- A *data model* consists of a set of concepts and languages for describing:
  - The conceptual and external schema
    - *Data definition language* (DDL)
  - The integrity constraints and domains (also DDL)
  - The operations on data
    - *Data manipulation language* (DML)
  - The directives that influence the physical schema (affects performance, not semantics)
    - *Storage definition language* (SDL)

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

The Relational Model

- A particular way of structuring data (using relations) proposed in 1970 by E. F. Codd
  - Mathematically based
    - Expressions (≡ *queries*) can be analyzed by DBMS
    - Queries are transformed to equivalent expressions automatically (*query optimization*)
      - Optimizers have limits (=> programmer needs to know how queries are evaluated and optimized)

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Relation Instance

- A *relation instance* is a **set** of tuples:
  - Tuple ordering immaterial
  - No duplicates
  - *Cardinality* of relation = number of tuples
- All tuples in a relation have the same structure; constructed from the same set of *attributes*:
  - Attributes are named (ordering is immaterial)
  - Value of an attribute is drawn from the attribute's *domain*.
  - *Arity* of relation = number of attributes.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Relation Instance

- There is also a special value ***NULL*** (commonly referred to as a *null value*, *value unknown* or *undefined*), which we use a placeholder and store it in place of an attribute until more information becomes available
  - null values arise because of a lack of information
  - null values arise by design sometimes
    - For instance, the attribute MaidenName is applicable to females but not to males

Employee(Id:INT, Name:STRING, MaidenName:STRING)

- We do not allow null values in certain sensitive places, such as the primary key.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Relation Instance of the Student relation

A relation instance can be represented as a table: the attributes are the column headers and the tuples are the rows:

Id	*Name*	*Address*	*Status*
1111111	John	123 Main	freshman
2345678	Mary	456 Cedar	sophmore
4433322	Art	77 So. 3rd	senior
7654321	Pat	88 No. 4th	sophmore

Student

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Relation Schema

- A relation schema is the heading of that table and the applicable constraints (integrity constraints).
- A *relation schema* is composed of:
  - Relation name
  - Attribute names & domains
  - *Integrity constraints* like:
    - The values of a particular attribute in all tuples are unique (e.g., PRIMARY KEY)
    - The values of a particular attribute in all tuples are greater than 0 (although, the domain can be bigger)
  - Default values for some attributes
- A *relation* consists of a relation schema and a relation instance.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Relational Database

- A *database schema* = the set of relation schemas and constraints among relations (*inter-relational* constraints)
- A *relational database* (or *database instance*) = a set of (corresponding) relation instances

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Database Schema Example

- *Student (Id: INT, Name: STRING, Address: STRING, Status: STRING)*
- *Professor (Id: INT, Name: STRING, DeptId: STRING)*
- *Course (DeptId:STRING, CrsCode:STRING, CrsName:STRING, Descr:STRING)*
- *Transcript (StudId:INTEGER, CrsCode:STRING, Semester:STRING, Grade:STRING)*
- *Department(DeptId: STRING, Name: STRING)*
- *Teaching (ProfId:INTEGER, CrsCode:STRING, Semester:STRING)*

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Integrity Constraints

- An *integrity constraint* (IC) is a statement about all legal instances of a database:
  - Restriction on a state (or of sequence of states) of a database (it is a part of the schema)
- Enforced/checked automatically by the DBMS
  - Protects database from errors
  - Enforces enterprise rules
- *Intra-relational* - involve only one relation
  - e.g., all Student Ids are unique (*key*)
- *Inter-relational* - involve several relations
  - e.g., the value of the attribute Id of each professor shown as Teaching a course must appear as the value of the Id attribute of some row of the table Professor (*foreign-key*)

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Kinds of Integrity Constraints

- Static – restricts legal states of database
  - Syntactic (structural constraints)
    - e.g., all values in a column must be unique
  - Semantic (involve meaning of attributes)
    - e.g., cannot register for more than 18 credits
- Dynamic constraints – limitation on sequences of database states
  - e.g., cannot raise salary by more than 5% in one transaction

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Kinds of Integrity Constraints

- Syntactic constraint example:

```
CREATE TABLE  STUDENT  (
      Id              INTEGER,
      Name            CHAR(20)   NOT NULL,
      Address         CHAR(50),
      Status          CHAR(10) DEFAULT 'freshman',
      PRIMARY KEY (Id) )
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Key Constraint

- A *key constraint* (or *candidate key*) is a sequence of attributes $A_1,\ldots,A_n$ (n=1 is possible) of a relation schema, S, with the following property:
  - A relation instance s of S satisfies the key constraint iff at most one row in s can contain a particular set of values, $a_1,\ldots,a_n$, for the attributes $A_1,\ldots,A_n$
  - Minimality property: no subset of $A_1,\ldots,A_n$ is a key constraint
- Key examples:
  - Set of attributes mentioned in a key constraint
    - e.g., {Id} in Student,
    - e.g., {StudId, CrsCode, Semester} in Transcript
  - It is minimal: no subset of a key is a key
    - {Id, Name} is not a key of Student

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Key Constraint

- Student(Id:INTEGER, Name:STRING, Address:STRING, Status:STRING)
  - Key: {Id}
- Professor(Id:INTEGER, Name:STRING, DeptId:STRING)
  - Key: {Id}
- Course(CrsCode:STRING, DeptId:STRING, CrsName:STRING, Descr:STRING)
  - Keys: {CrsCode}, {DeptId,CrsName}
- Transcript(StudId:INTEGER, CrsCode:STRING, Semester:STRING, Grade:STRING)
  - Key: {StudId,CrsCode,Semester}
- Teaching(ProfId:INTEGER, CrsCode:STRING, Semester:STRING)
  - Key: {CrsCode,Semester}

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Key Constraint

- One of the keys of a relation is designated as the *primary key*.

- Superkey - set of attributes containing key
  - {Id, Name} is a superkey of Student

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Key Constraint

- Every relation has a key:
  - the set of all attributes in a schema, S, is always a superkey because if a legal instance of S has a pair of tuples that agree on all attributes in S, then these must be identical tuples: since relations are sets, they cannot have identical elements

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Key Constraint

- A schema can have several different keys.
  - Example:
    - in the Course relation:
      - {CrsCode} can be one key.
      - {DeptId, CrsName} is also a key in the same relation because because it is unlikely that the same department will offer two different courses with the same name.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Foreign Key Constraint

- *Referential integrity*: an item named in one relation must refer to tuples that describe that item in another
  - Transcript(CrsCode) references Course(CrsCode)
  - Professor(DeptId) references Department(DeptId)

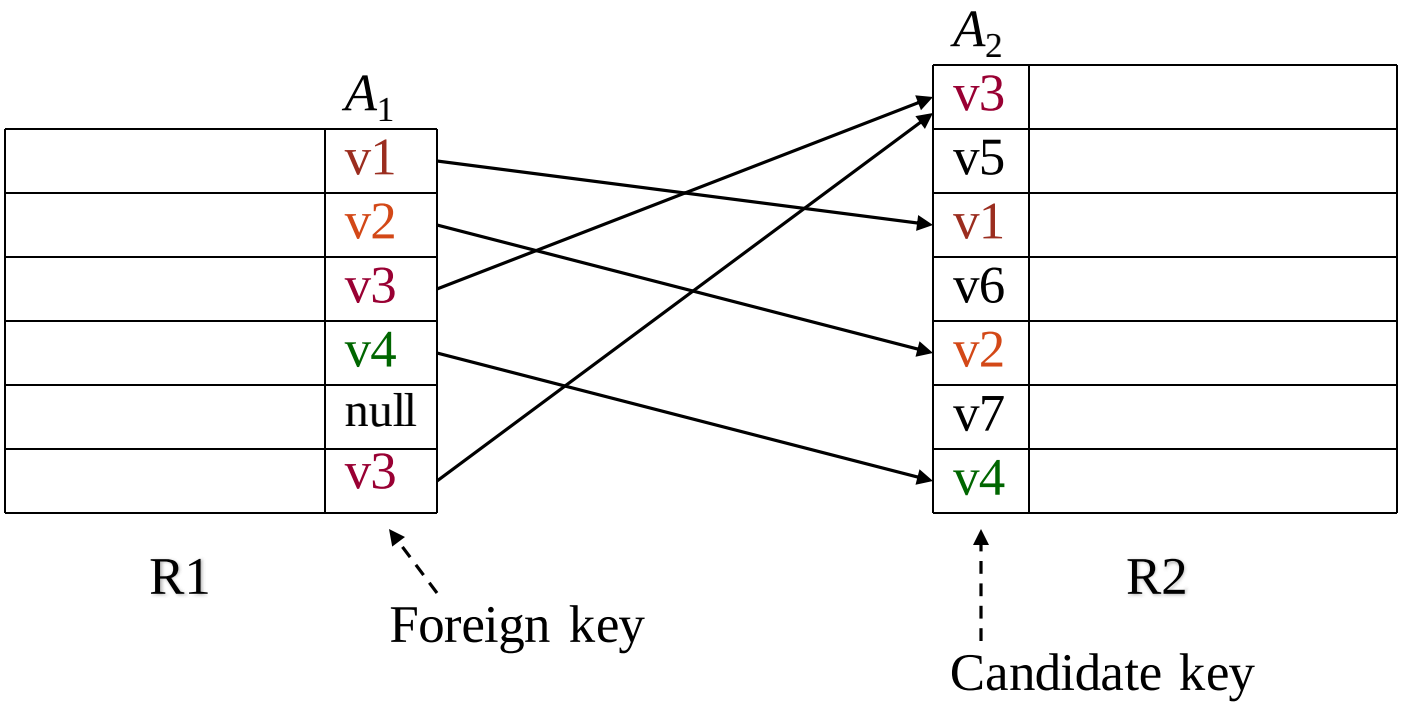
(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Foreign Key Constraint

- Attribute A_1 is a *foreign key* of R_1 referring to attribute A_2 in R_2, if whenever there is a value v of A_1, there is a tuple of R_2 in which A_2 has value v, and A_2 is a key of R_2
  - This is a special case of referential integrity: A_2 must be a candidate key of R_2 (e.g., CrsCode is a key of Course relation)
  - If no row exists in R_2 => *violation of referential integrity*
  - Not all rows of R_2 need to be referenced: relationship is not symmetric (e.g., some course might not be taught)
  - Value of a foreign key might not be specified (DeptId column of some professor might be NULL)

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Foreign Key Constraint



(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Foreign Key Constraint

- Names of the attrs. A_1 and A_2 need not be the same

Teaching(*CrsCode*: COURSES, *Sem*: SEMESTERS, *ProfId*: INT)

Professor(*Id*: INT, *Name*: STRING, *DeptId*: DEPTS)

  - ProfId attribute of Teaching references Id attribute of Professor

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Foreign Key Constraint

- R_1 and R_2 need not be distinct

  Employee(Id:INT, MgrId:INT, ….)
  - Employee(MgrId) references Employee(Id)
    - Every manager is also an employee and hence has a unique row in Employee

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Foreign Key Constraint

- Foreign key might consist of several columns
  - $R_1(A_1, \ldots A_n)$ references $R_2(B_1, \ldots B_n)$
    - A_i and B_i must have same domains (although not necessarily the same names)
    - $B_1, \ldots, B_n$ must be a candidate key of R_2
  - Example: (CrsCode, Semester) of Transcript references (CrsCode, Semester) of Teaching

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Inclusion Dependency

- Referential integrity constraint that is not a foreign key constraint
  - Example: Teaching(CrsCode, Semester) references Transcript(CrsCode, Semester)
    - No empty classes allowed
    - Target attributes do not form a candidate key in Transcript (StudId missing)
- No simple enforcement mechanism for inclusion dependencies in SQL (requires assertions)

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

SQL Data Definition Sublanguage

- SQL is a language for describing database schema and operations on tables
- Data Definition Language (DDL): sublanguage of SQL for describing schema

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

SQL

- An SQL schema is a description of a portion of a database

```
CREATE SCHEMA SRS_StudInfo
```

- It usually is under the control of a single user who has the authorization to create and access the objects within it.

```
CREATE SCHEMA SRS_StudInfo AUTHORIZATION Joe
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

SQL

- Joe can also delete the schema:

```
DROP SCHEMA SRS_StudInfo
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Tables

- Table: is an SQL entity that corresponds to a relation
  - SQL-92 is currently the most supported standard but is now superseded by SQL:1999 (recursive queries, triggers, and some object-oriented features), SQL:2003 (SQL/XML), SQL:2006 (storing XML data in an SQL, xQuery), SQL:2008 (ORDER BY), SQL:2011

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Table Declaration

```
CREATE TABLE Student (
   Id  INTEGER,
   Name  CHAR(20),
   Address  CHAR(50),
   Status  CHAR(10)
)
```

Id	*Name*	*Address*	*Status*
101222333	John	10 Cedar St	Freshman
234567890	Mary	22 Main St	Sophomore

Student

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

SQL Types

- Database vendors generally deviate from the standard even on basic features like the data types:
  - SQL Data Types for Various DBs:

    http://www.w3schools.com/sql/sql_datatypes.asp
- We will focus on MySQL here.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

SQL CHAR Types

- CHAR(n)
  - Used to store character string value of fixed length.
  - The maximum no. of characters the data type can hold is 255 characters.
  - It's 50% faster than VARCHAR.
  - Uses static memory allocation.
  - Good for things of the same size (e.g., signatures, keys, zips, phone, ssn)
- VARCHAR(n)
  - Used to store variable length alphanumeric data.
  - The maximum this data type can hold is up to
    - Pre-MySQL 5.0.3: 255 characters.
    - In MySQL 5.0.3+: 65,535 characters shared for the row.
  - It's slower than CHAR.
  - Uses dynamic memory allocation.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

SQL Numeric Types

- INTEGER (INT)
  - Integer numerical (no decimal). Precision 10.
- SMALLINT
  - Integer numerical (no decimal). Precision 5
- BIGINT
  - Integer numerical (no decimal). Precision 19
- DECIMAL(p,s) (same with NUMERIC, DOUBLE)
  - Exact numerical, precision p, scale s. Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal
- REAL
  - Approximate numerical, mantissa precision 7
- FLOAT
  - Approximate numerical, mantissa precision 16

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

SQL TIME Types

- DATE
  - Stores year, month, and day values.
  - Format: YYYY-MM-DD.
  - The supported range is from '1000-01-01' to '9999-12-31'.
- TIME
  - Stores hour, minute, and second values
  - Format: HH:MI:SS.
  - The supported range is from '-838:59:59' to '838:59:59'.
- DATETIME
  - A date and time combination.
  - Format: YYYY-MM-DD HH:MI:SS
- TIMESTAMP
  - Stores year, month, day, hour, minute, and second values

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Other SQL Types

- BLOB
  - For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
- ENUM(x,y,z,etc.)
  - Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted.
- SET
  - Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice.
- XML
  - Stores XML formatted data.
  - Not in MySQL, but in DB2, Oracle, SQL Server.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Primary/Candidate Keys

```
CREATE TABLE Course (
   CrsCode CHAR(6),
   CrsName CHAR(20),
   DeptId CHAR(4),
   Descr CHAR(100),
   PRIMARY KEY (CrsCode),
   UNIQUE (DeptId, CrsName)   -- candidate key
)
```

Comments start with 2 dashes

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Null

- Problem: Not all information might be known when row is inserted (e.g., a Grade might be missing from Transcript)
- A column might not be applicable for a particular row (e.g., MaidenName if row describes a male)
- Solution: Use place holder: null
  - Not a value of any domain (although called null value)
    - Indicates the absence of a value
  - Not allowed in certain situations
    - Primary keys and columns constrained by NOT NULL

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Default Value

- Value to be assigned if attribute value in a row is not specified

```
CREATE TABLE Student (
   Id INTEGER,
   Name CHAR(20) NOT NULL,
   Address CHAR(50),
   Status CHAR(10) DEFAULT 'freshman',
   PRIMARY KEY (Id)
)
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Semantic Constraints in SQL

- Primary key and foreign key are examples of structural constraints
- Semantic constraints
  - Express the logic of the application at hand:
    - e.g., number of registered students ≤ maximum enrollment

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Semantic Constraints in SQL

- Often used for application dependent conditions
- Example: limit attribute values

```
CREATE TABLE Transcript (
    StudId  INTEGER,
    CrsCode  CHAR(6),
    Semester  CHAR(6),
    Grade  CHAR(1),
    CHECK (Grade IN ('A', 'B', 'C', 'D', 'F')),
    CHECK (StudId > 0 AND StudId < 1000000000)
)
```

- Each row in table must satisfy condition

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Semantic Constraints in SQL

- Example: relate values of attributes in different columns

```
CREATE TABLE Employee (
    Id   INTEGER,
    Name   CHAR(20),
    Salary   INTEGER,
    MngrSalary   INTEGER,
    CHECK ( MngrSalary > Salary)
)
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Constraints – Problems

- Problem 1:
  - The semantics of the CHECK clause requires that every tuple in the corresponding relation satisfy all of the conditional expressions associated with all CHECK clauses in the corresponding CREATE TABLE statement
  - The empty relation (i.e., a relation that contains no tuples) always satisfies all CHECK constraints as there are no tuples to check.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Constraints – Problems

- If Employee is empty, there are no rows on which to evaluate the CHECK condition.

```
CREATE TABLE Employee (
   Id  INTEGER,
   Name  CHAR(20),
   Salary  INTEGER,
   DepartmentId  CHAR(4),
   MngrId  INTEGER,
   CHECK ( 0 < (SELECT COUNT(*)  FROM Employee))
   CHECK ( SELECT COUNT(*) FROM Manager)
          < (SELECT COUNT(*)FROM  Employee)  ) )
```

  - The first CHECK clause says that the Employee relation cannot be empty
    - The conditions are supposed to be satisfied by every tuple in the Employee relation, not by the relation itself

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Constraints – Problems

```
CREATE TABLE Employee (
    Id  INTEGER,
    Name  CHAR(20),
    Salary  INTEGER,
    DepartmentId  CHAR(4),
    MngrId  INTEGER,
    CHECK ( 0 < (SELECT COUNT (*) FROM Employee))
    CHECK ( SELECT COUNT(*) FROM Manager)
          < (SELECT COUNT(*)FROM Employee)  ) )
```

- The second CHECK clause says that there must be more employees than managers, which it in fact does, but only if the Employee relation is not empty.
  - Should constraint be in Employee or in Manager?
  - Inter-relational constraints should be symmetric.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Assertion

- Symmetrically specifies an inter-relational constraint
- Applies to entire database (not just the individual rows of a single table)
  - hence it works even if Employee is empty
- Unlike the CHECK conditions that appear inside a table definition, those in the CREATE ASSERTION statement must be satisfied by the contents of the entire database rather than by individual tuples of a host table.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Assertion

- Example: the Employee table cannot be empty:

```
CREATE ASSERTION EmployeeNotEmpty
  CHECK (0 < SELECT  COUNT (*)  FROM Employee)
```

- If, at the time of specifying the constraint, the Employee relation is empty, the SQL standard states that if a new constraint is defined and the existing database does not satisfy it, the constraint is rejected.
  - The database designer then has to find out the cause of constraint violation and either rectify the database or amend the constraint.
- MySQL does not support ASSERTIONS.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Assertion

- Example: less managers than employees:

```
CREATE ASSERTION EmployeesMoreThanManagers
  CHECK (SELECT COUNT(*) FROM Manager)
        < (SELECT COUNT(*) FROM Employee) ))
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Assertion

- Example: employees cannot earn more than their managers:

```
CREATE ASSERTION EmployeeSalariesDown
    CHECK NOT EXISTS
        (SELECT * FROM Employee, Manager
        WHERE Employee.MngrId = Manager.Id AND
                Employee.Salary > Manager.Salary ))
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Assertions and Inclusion Dependency

No tuple in the Teaching relation (the outer NOT EXISTS statement) for which no matching class exists in the Transcript relation (the inner NOT EXISTS statement).

```
CREATE ASSERTION NoEmptyCourses
  CHECK (NOT EXISTS (
            SELECT * FROM Teaching T
            WHERE  -- for each row T check
                   -- the following condition
              NOT EXISTS (
              SELECT * FROM Transcript R
              WHERE T.CrsCode = R.CrsCode
                AND T.Semester = R.Semester)
  ))
```

Courses with no students

Students in a particular course

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

User-Defined Domains

- Possible attribute values can be specified
  - Using a CHECK constraint or
  - Creating a new domain
    - An alternative SQL way to allow the user to define appropriate ranges of values, give them domain names, and then use these names in various table definitions.
  - Domain can be used in several declarations

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Domains

- Examples:

```
CREATE DOMAIN Grades CHAR (1)
   CHECK  (VALUE IN ('A', 'B', 'C', 'D', 'F'))

CREATE TABLE Transcript (
    ….,
    Grade: Grades,
    …  )
```

- MySQL doesn't support domains, but if you want to create them, then you can use MariaDB.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

SQL Foreign Key Constraints

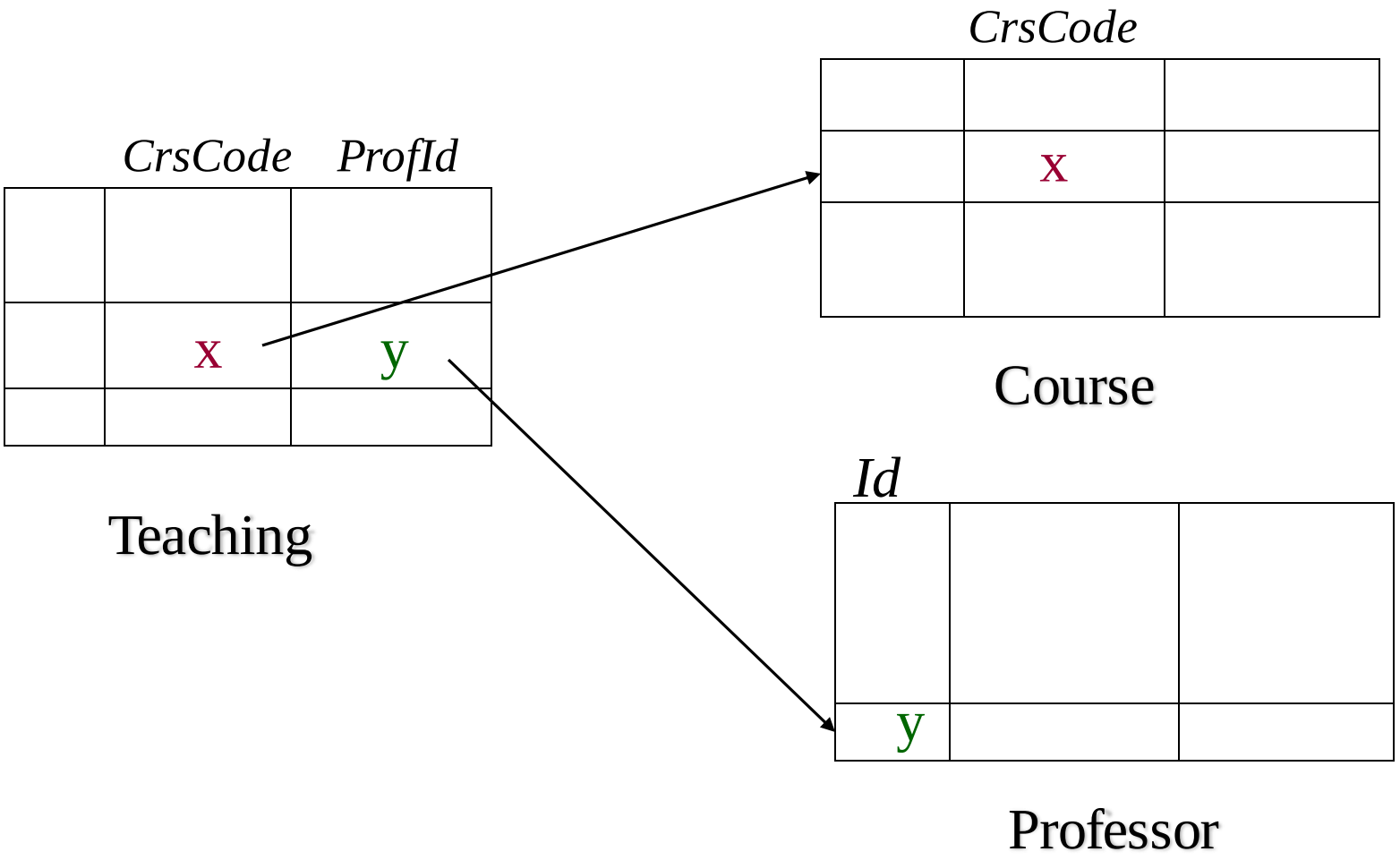
```
CREATE TABLE Teaching (
    ProfId   INTEGER,
    CrsCode   CHAR (6),
    Semester   CHAR (6),
    PRIMARY KEY (CrsCode, Semester),
    FOREIGN KEY (CrsCode) REFERENCES Course,
    FOREIGN KEY (ProfId) REFERENCES Professor (Id) )
```

- The Course table must have a CrsCode attribute and key.
- In MySQL, we must specify the attribute even if it is the same name.

```
FOREIGN KEY (CrsCode) REFERENCES Course(CrsCode)
```

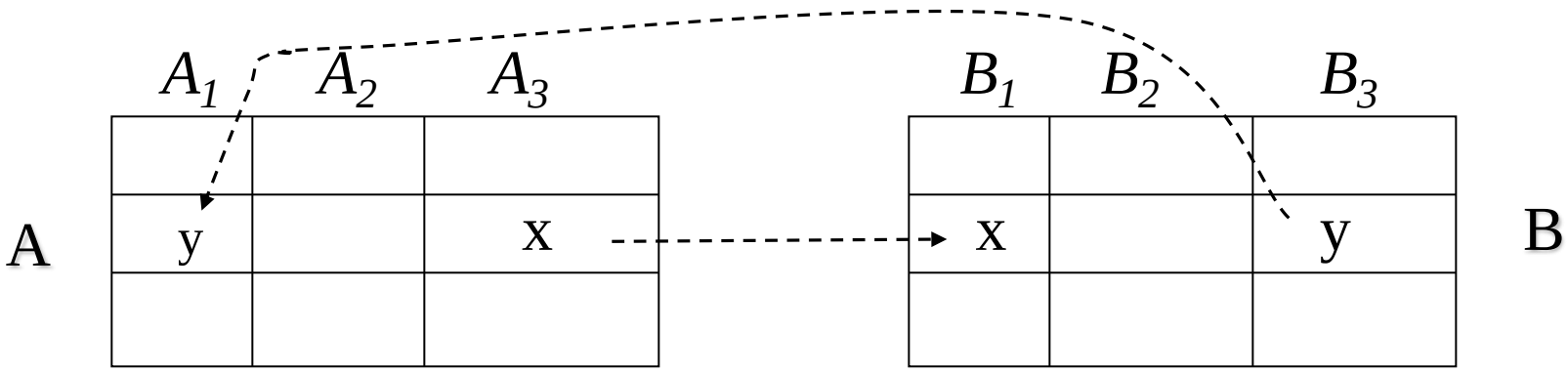

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Foreign Key Constraints



(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Circularity in Foreign Key Constraints



candidate key: A_1
foreign key: A_3 references B(B_1)

candidate key: B_1
foreign key: B_3 references A(A_1)

- Chicken-and-egg problem:

Problem 1: Creation of A requires existence of B and vice versa

Solution:

```
CREATE TABLE A ( ……)    -- no foreign key
CREATE TABLE B ( ……)    -- include foreign key
ALTER TABLE A
   ADD CONSTRAINT cons
      FOREIGN KEY (A3) REFERENCES B (B1)
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Circularity in Foreign Key Constraints

- Problem 2: Insertion of row in A requires prior existence of row in B and vice versa

Solution: use appropriate constraint checking mode:

- IMMEDIATE checking: a check is made after each SQL statement that changes the database
- DEFERRED checking: a check is made only when a transaction commits.

```
ALTER TABLE chicken ADD CONSTRAINT chickenREFegg
    FOREIGN KEY (eID) REFERENCES egg(eID)
    INITIALLY DEFERRED DEFERRABLE;
ALTER TABLE egg ADD CONSTRAINT eggREFchicken
    FOREIGN KEY (cID) REFERENCES chicken(cID)
    INITIALLY DEFERRED DEFERRABLE;
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Reactive Constraints

- When a constraint is violated, the corresponding transaction is typically aborted.
  - However, in some cases, other remedial actions are more appropriate.
    - Foreign-key constraints are one example of this situation.
- It would be nice to have a mechanism that allows a user to specify how to react to a violation of a constraint.
  - A reactive constraint is a static constraint coupled with a specification of what to do if a certain event happens.
    - Triggers attached to foreign-key constraints

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

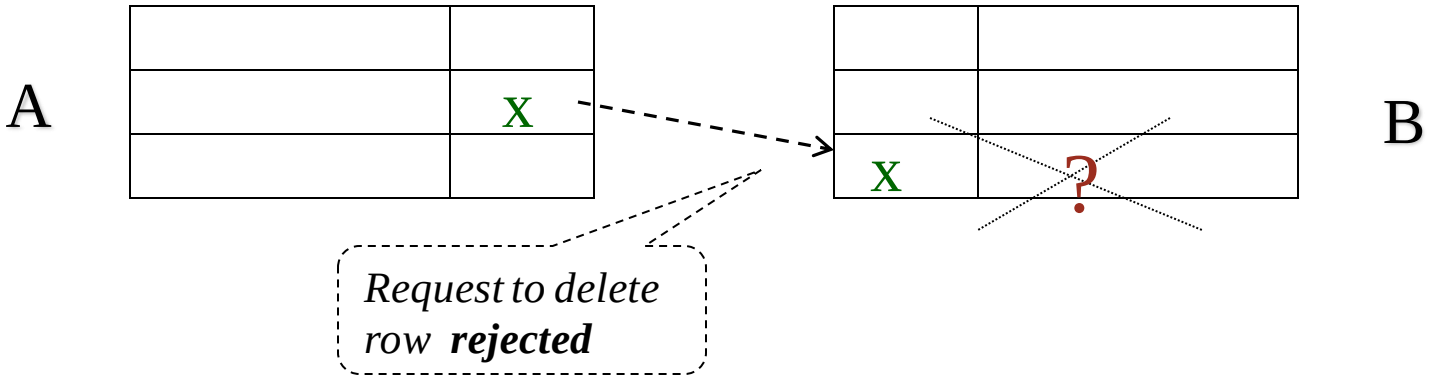
Handling Foreign Key Violations

- Insertion into A:
  - Reject if no row exists in B containing foreign key of inserted row!

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Handling Foreign Key Violations

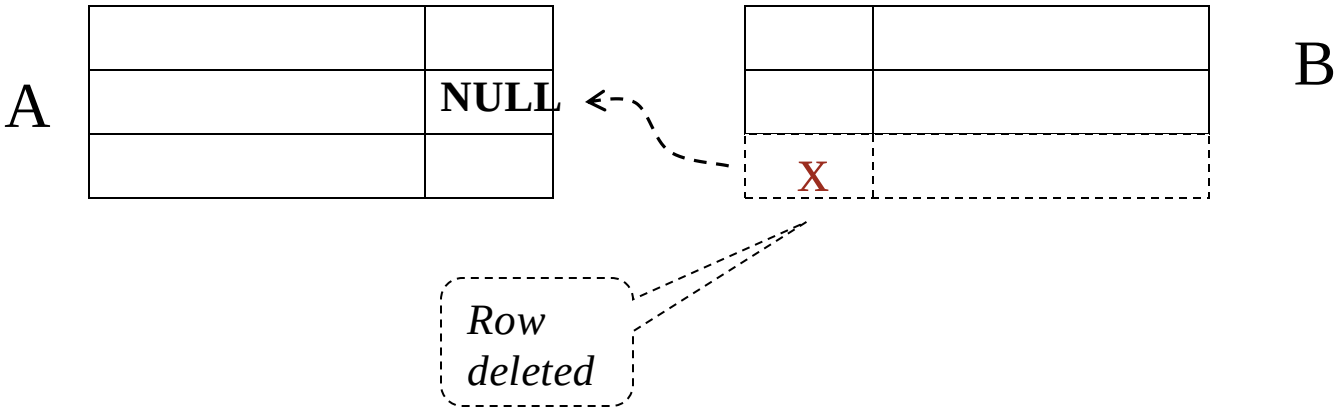
- Deletion from B:
  - Multiple possible responses.
  - NO ACTION: Reject if row(s) in A references row to be deleted (default response)



(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Handling Foreign Key Violations

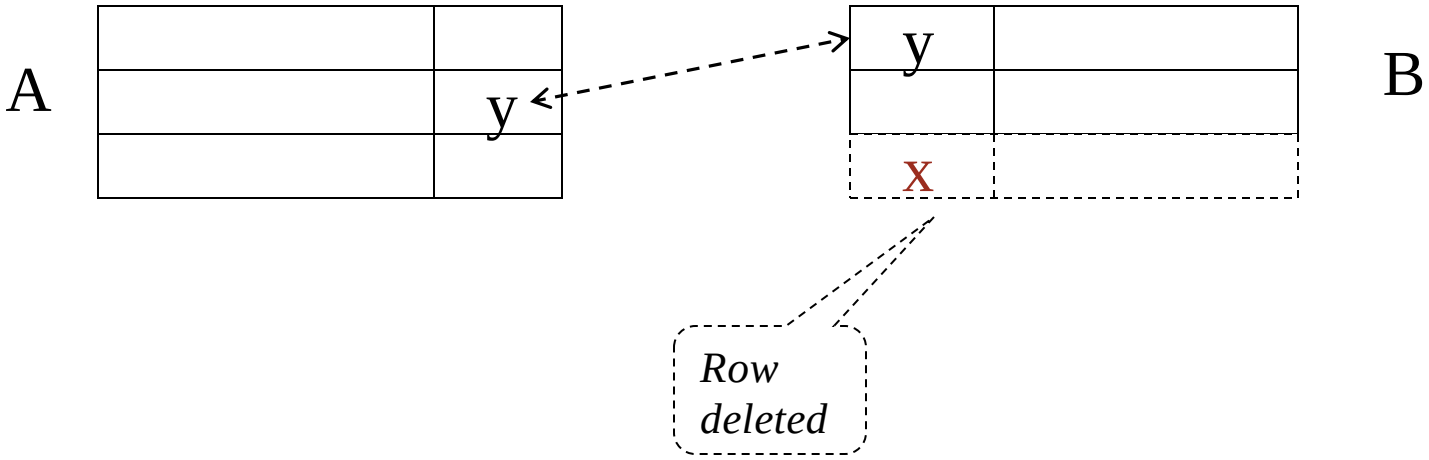
- Deletion from B:
  - SET NULL: Set value of foreign key in referencing row(s) in A to NULL



(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Handling Foreign Key Violations

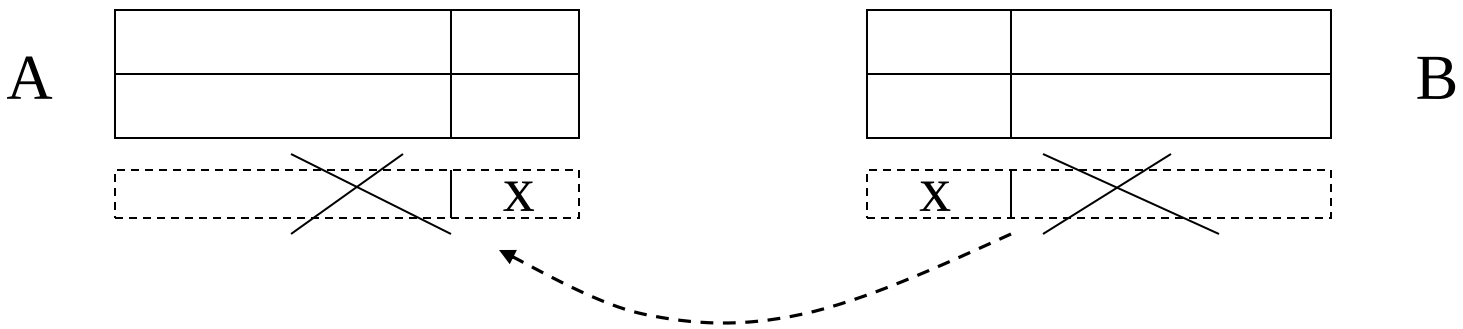
- Deletion from B:
  - SET DEFAULT: Set value of foreign key in referencing row(s) in A to default value (y) which must exist in B



(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Handling Foreign Key Violations

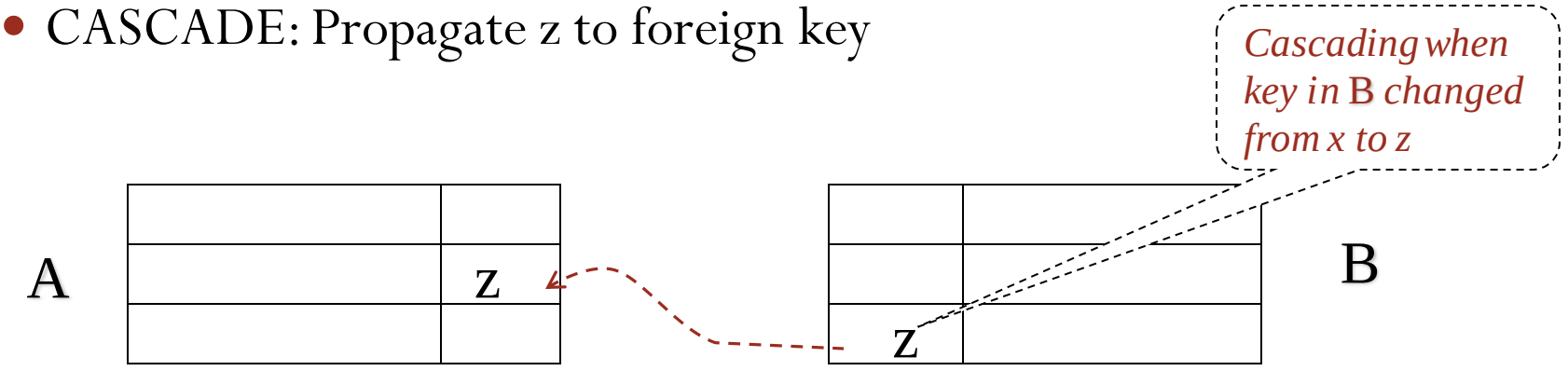
- Deletion from B:
  - CASCADE: Delete referencing row(s) in A as well



(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Handling Foreign Key Violations

- Update (change) foreign key in A: Reject if no row exists in B containing new foreign key
- Update candidate key in B (to z) – same actions as with deletion:
  - NO ACTION: Reject if row(s) in A references row to be updated (default response)
  - SET NULL: Set value of foreign key to null
  - SET DEFAULT: Set value of foreign key to default
  - CASCADE: Propagate z to foreign key



(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

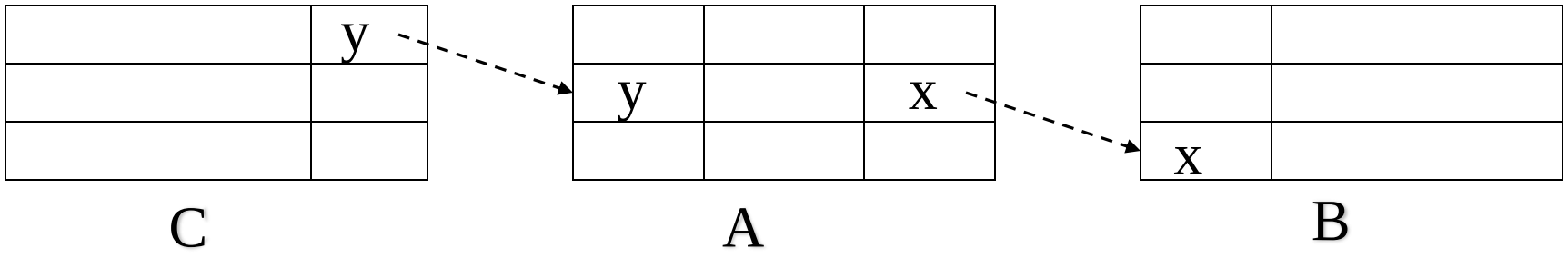
Specifying Actions

```
CREATE TABLE Teaching (
    ProfId     INTEGER,
    CrsCode CHAR (6),
    Semester CHAR (6),
    PRIMARY KEY (CrsCode, Semester),

    FOREIGN KEY (ProfId) REFERENCES Professor (Id)
       ON DELETE NO ACTION
       ON UPDATE CASCADE,

    FOREIGN KEY (CrsCode) REFERENCES Course (CrsCode)
       ON DELETE NO ACTION
       ON UPDATE CASCADE )
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Handling Foreign Key Violations

- The action taken to repair the violation of a foreign key constraint in A may cause a violation of a foreign key constraint in C
  - The action specified in C controls how that violation is handled
  - If the entire chain of violations cannot be resolved, the initial deletion from B is rejected



(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Triggers

- The ON DELETE/UPDATE triggers are simple and powerful, but they are not powerful enough to capture a wide variety of constraint violations that arise in database applications and are not due to foreign keys
- A more general mechanism for handling events
  - Whenever a specified event occurs, execute some specified action
  - Not in SQL-92, but is in SQL:1999

```
CREATE TRIGGER CrsChange
    AFTER UPDATE OF CrsCode, Semester ON Transcript
    WHEN  (Grade IS NOT NULL)
        ROLLBACK
```

- Trigger is a schema element (like table, assertion, etc.)

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Insert

- Inserting a single row into a table
  - Attribute list can be omitted if it is the same as in CREATE TABLE (but do not omit it)
  - NULL and DEFAULT values can be specified

```
INSERT INTO Student(Id, Name, Address, Status)
VALUES (12345, 'John Smith', '123 Main St, NYC', NULL)
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Database Views

- Part of external schema
- A virtual table constructed from actual tables on the fly
  - Can be accessed in queries like any other table
  - Not materialized, constructed when accessed
  - Similar to a subroutine in ordinary programming

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Views - Examples

- Part of external schema suitable for use in Bursar's office:

```
CREATE VIEW CoursesTaken (StudId, CrsCode, Semester) AS
    SELECT  T.StudId, T.CrsCode, T.Semester
    FROM Transcript T
```

- Part of external schema suitable for student with Id 123456789:

```
CREATE VIEW CoursesITook (CrsCode, Semester, Grade) AS
     SELECT  T.CrsCode, T.Semester, T.Grade
     FROM  Transcript T
     WHERE  T.StudId = 123456789
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Modifying the Schema

- Although database schemas are not supposed to change frequently, they do evolve (new fields are added, etc.)

```
ALTER TABLE Student
   ADD COLUMN  Gpa  INTEGER DEFAULT 0

ALTER TABLE Student
   ADD CONSTRAINT GpaRange
      CHECK (Gpa >= 0 AND Gpa <= 4)

ALTER TABLE  Student
   DROP CONSTRAINT GpaRange    -- constraint names are useful

DROP TABLE  Employee

DROP ASSERTION  NoEmptyCourses
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Access Control

- Databases might contain sensitive information
- Access has to be limited:
  - Users have to be identified – authentication
    - Generally done with passwords
  - Each user must be limited to modes of access appropriate to that user - authorization
- SQL:92 provides tools for specifying an authorization policy but does not support authentication (i.e., vendor specific)

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Controlling Authorization in SQL

```
CREATE USER 'joe'@'localhost' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON * . * TO 'newuser'@'localhost';
```


(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Controlling Authorization in SQL

GRANT *access_list*
ON *table*
TO *user_list*

access modes: SELECT, INSERT, DELETE, UPDATE, REFERENCES, All PRIVILEGES

GRANT UPDATE (*Grade*) ON Transcript TO prof_smith

User name

– The *Grade* column can be updated only by prof_smith

GRANT SELECT ON Transcript TO joe

– Individual columns cannot be specified for SELECT access (in the SQL standard) – all columns of Transcript can be read
– *But* SELECT access control to individual columns can be *simulated* through views

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Controlling Authorization in SQL Using Views

GRANT *access*
ON *view*
TO *user_list*

GRANT SELECT ON Course Taken TO joe

- Thus views can be used to simulate access control to individual columns of a table

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

REVOKE

- Privileges, or the grant option for privileges, can be revoked using the REVOKE statement.

  REVOKE [GRANT OPTION FOR] privilege-list
  ON object
  FROM user-list {CASCADE | RESTRICT}

  - CASCADE means that if some user, U_1, whose user name appears on the list userlist, has granted those privileges to another user, U_2, the privileges granted to U2 are also revoked.
    - If U2 has granted those privileges to still another user, those privileges are revoked as well, and so on.
  - RESTRICT means that if any such dependent privileges exist, the REVOKE statement is rejected

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

REFERENCE Access mode

- The GRANT REFERENCES permission on a table is needed to create a FOREIGN KEY constraint that references that table.

GRANT REFERENCES ON Student TO joe

- Now Joe can create tables that have foreign keys to Student keys

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

Access mode problem

- Granting privileges at the level of database operations, such as SELECT or UPDATE, is not adequate.
  - For example, only a depositor can deposit in a bank account and only a bank official can add interest to the account, but both the deposit and interest transactions might use the same UPDATE statement.
- For such applications, it is more appropriate to grant privileges at the level of subroutines or transactions.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)