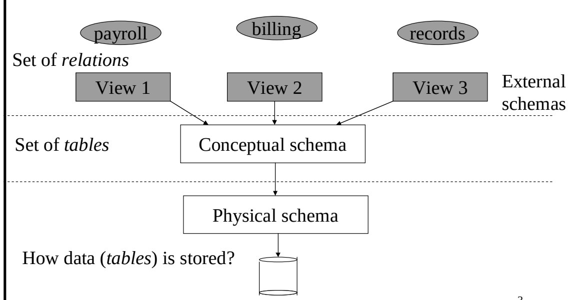
The Relational Data Model

Chapter 4

Data and Its Structure

- Data is actually stored as bits, but it is difficult to work with data at this level.
- It is convenient to view data at different *levels of abstraction*.
- ***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*.

Three Schemas in the Relational Data Model



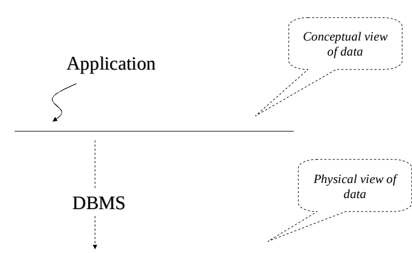
Physical Data Level

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

Conceptual Data Level

- Hides details.
  - In the relational model, the conceptual schema presents data as a set of tables.
- DBMS maps from conceptual to physical schema automatically.
- 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***

Conceptual Data Level (con't)



External Data Level

- 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. 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 files in full.
    - 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.

7

External Data Level (con't)

- Application is written in terms of an external schema.
- A view is computed when accessed (not stored).
- Different external schemas can be provided to different categories of users.
- Translation from external to conceptual done automatically by DBMS at run time.
- Conceptual schema can be changed without changing application:
  - Mapping from external to conceptual must be changed.
- Referred to as ***conceptual data independence***.

8

Data Model

- Tools and language for describing:
  - Conceptual and external schema (a schema: description of data at some level (*e.g.*, tables, attributes, constraints, domains)
    - *Data definition language* (DDL)
  - Integrity constraints, domains (DDL)
  - Operations on data
    - *Data manipulation language* (DML)
  - *Optional*: Directives that influence the physical schema (affects performance, not semantics)
    - *Storage definition language* (SDL)

SQL

9

Relational Model

- A particular way of structuring data (using relations)
- Simple
- 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)

10

Relation Instance

- Relation 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*
    - There is also a special value **null** (value unknown or undefined), which belongs to no domain
  - *Arity* of relation = number of attributes

11

Relation Instance (Example)

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

12

Relation Schema

- Relation name
- Attribute names & domains
- Integrity constraints like
  - The values of a particular attribute in all tuples are unique
  - The values of a particular attribute in all tuples are greater than 0
- Default values

13

Relational Database

- Finite set of relations
- Each relation consists of a schema and an instance
- *Database schema* = set of relation schemas constraints among relations (*inter-relational* constraints)
- *Database instance* = set of (corresponding) relation instances

14

Database Schema (Example)

- Student (*Id*: INT, *Name*: STRING, *Address*: STRING, *Status*: STRING)
- Professor (*Id*: INT, *Name*: STRING, *DeptId*: DEPTS)
- Course (*DeptId*: DEPTS, *CrsName*: STRING, *CrsCode*: COURSES)
- Transcript (*CrsCode*: COURSES, *StudId*: INT, *Grade*: GRADES, *Semester*: SEMESTERS)
- Department(*DeptId*: DEPTS, *Name*: STRING)

15

Integrity Constraints

- Part of schema
- Restriction on state (or of sequence of states) of data base
- Enforced by DBMS
- *Intra-relational* - involve only one relation
  - Part of relation schema
  - e.g., all *Id*s are unique
- *Inter-relational* - involve several relations
  - Part of relation schema or database schema

16

Constraint Checking

- <u>Automatically</u> checked by DBMS
- Protects database from errors
- Enforces enterprise rules

17

Kinds of Integrity Constraints

- Static – restricts legal states of database
  - Syntactic (structural)
    - 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 – limitation on sequences of database states
    - e.g., cannot raise salary by more than 5%

18

Key Constraint

- A *key constraint* is a sequence of attributes $A_1,\ldots,A_n$ (n=1 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*: no subset of $A_1,\ldots,A_n$ is a key constraint
- ***Key***
  - 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

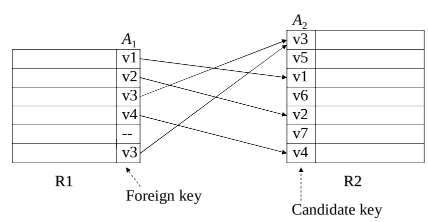
Key Constraint (cont'd)

- ***Superkey*** **-** set of attributes containing key
  - (*Id*, *Name*) is a superkey of Student
- Every relation has a key
- Relation can have several keys:
  - ***primary key:*** *Id* in Student (can't be **null**)
  - ***candidate key*:** (*Name*, *Address*) in Student

Foreign Key Constraint

- ***Referential integrity:*** 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*)
- Attribute A_1 is a ***foreign key*** of R1 referring to attribute A_2 in R2, if whenever there is a value *v* of A_1, there is a tuple of R2 in which A_2 has value *v*, and A_2 is a key of R2
  - This is a special case of referential integrity: A_2 must be a candidate key of R2 (e.g., *CrsCode* is a key of Course in the above)
  - If no row exists in R2 => violation of referential integrity
  - Not all rows of R2 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**)

Foreign Key Constraint (Example)



Foreign Key (cont'd)

- Names of A_1 and A_2 need not be the same.
  - With tables:

    Teaching(*CrsCode*: COURSES, *Sem*: SEMESTERS, *ProfId*: INT)
    Professor(*Id*: INT, *Name*: STRING, *DeptId*: DEPTS)

  ProfId attribute of Teaching references *Id* attribute of Professor
- R1 and R2 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

Foreign Key (cont'd)

- Foreign key might consist of several columns
  - (*CrsCode*, *Semester*) of Transcript references (*CrsCode*, *Semester*) of Teaching
- R1(A_1, …A_n) references R2(B_1, …B_n)
  - There exists a 1 - 1 correspondance between $A_1,\ldots A_n$ and $B_1,\ldots B_n$
  - A_i and B_i have same domains (although not necessarily the same names)
  - $B_1,\ldots,B_n$ is a candidate key of R2

Inclusion Dependency

- Referential integrity constraint that is not a foreign key constraint
- 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* -- *later*)

25

SQL

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

26

Tables

- SQL entity that corresponds to a relation
- An element of the database schema
- SQL-92 is currently the most supported standard but is now superseded by SQL:1999
- Database vendors generally deviate from standard, but eventually converge

27

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

28

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
)
```

Things that start with 2 dashes are comments

29

Null

- **Problem**: Not all information might be known when row is inserted (e.g., *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

30

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) )
```

31

Semantic Constraints in SQL

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

32