CSIE30600/CSIEB0290 Database Systems

Lecture 3: Relational Model

Outline

- Relational Model Concepts
- Relational Model Constraints
- Relational Database Schemas
- Update Operations
- Transactions
- Dealing with Constraint Violations

Data Model Revisited

- Provides the means for **specifying** particular **data structures**, for **constraining** the data sets associated with these structures, and for **manipulating** the data
- Data definition language (DDL): define structures and constraints
- Data manipulation language (DML): specify manipulations/operations over the data

Why Relational Model

- Extremely useful and simple
  – Single data-modeling concept: relations = 2-D tables
  – Allows clean yet powerful manipulation languages
- Most widely used model
  – Vendors: Oracle, IBM(DB2, Informix), Microsoft(SQL Server, Access), etc.
- Recent competitors: object-relational model, semi-structured model, document, key-value, NoSQL, NewSQL
  – MongoDB(document), Redis(key-value), Cassandra(NoSQL), HBase(NoSQL)
  – Object-oriented aspects of SQL:1999

DB Engine Ranking

395 systems in ranking, August 2022

Rank			DBMS	Database Model	Score		
Aug 2022	Jul 2022	Aug 2021			Aug 2022	Jul 2022	Aug 2021
1.	1.	1.	Oracle	Relational, Multi-model	1260.80	-19.50	-8.46
2.	2.	2.	MySQL	Relational, Multi-model	1202.85	+7.98	-35.37
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	944.96	+2.83	-28.39
4.	4.	4.	PostgreSQL	Relational, Multi-model	618.00	+2.13	+40.95
5.	5.	5.	MongoDB	Document, Multi-model	477.66	+4.68	-18.88
6.	6.	6.	Redis	Key-value, Multi-model	176.39	+2.77	+6.51
7.	7.	7.	IBM Db2	Relational, Multi-model	157.23	-3.99	-8.24
8.	8.	8.	Elasticsearch	Search engine, Multi-model	155.08	+0.75	-2.01
9.	9.	↑ 10.	Microsoft Access	Relational	146.50	+1.41	+31.66
10.	10.	↓ 9.	SQLite	Relational	138.87	+2.20	+9.06
11.	11.	11.	Cassandra	Wide column	118.15	+3.74	+4.49
12.	12.	12.	MariaDB	Relational, Multi-model	113.89	+1.37	+14.92
13.	13.	↑ 23.	Snowflake	Relational	103.12	+3.97	+56.58
14.	14.	↓ 13.	Splunk	Search engine	97.44	-0.76	+6.84
15.	↑ 16.	↑ 16.	Amazon DynamoDB	Multi-model	87.26	+3.32	+12.36
16.	↓ 15.	↓ 15.	Microsoft Azure SQL Database	Relational, Multi-model	86.18	+1.28	+11.02
17.	17.	↓ 14.	Hive	Relational	78.66	-0.82	-5.27
18.	18.	↓ 17.	Teradata	Relational, Multi-model	69.07	-1.85	+0.25
19.	19.	↓ 18.	Neo4j	Graph	59.35	+0.94	+2.40
20.	20.	20.	Solr	Search engine, Multi-model	55.78	+0.08	+4.71

Relational Model Concepts

- The relational model of data is based on the concept of a *relation*
  - The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations
- We review the essentials of the formal relational model in this chapter
- In ***practice***, there is a standard model based on SQL – to be described in later lectures
- Note: There are several important differences between the *formal* model and the *practical* model, as we shall see

Relational Model Concepts

- A relation is a mathematical concept based on sets
- The model was first proposed by Dr. E. F. Codd of IBM Research in 1970 in the following paper: (next slide)
  - "A Relational Model for Large Shared Data Banks," *Communications of the ACM*, June 1970.
  - use relations as data structures, algebra for specifying queries, no mechanisms for updates or constraints
  - follow-up papers introduced new language based on first-order logic and showed it is equivalent to the algebra, introduced integrity constraints
- These papers caused a major revolution in the field of database management and earned Dr. Codd the coveted ACM Turing Award

Dawn of the Relational Model

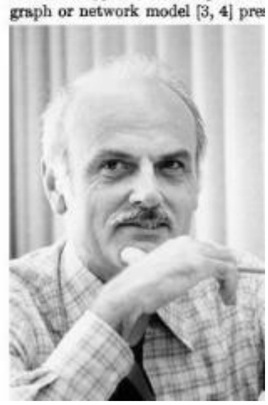
Information Retrieval

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.
Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network

The relational view (or model
Section 1 appears to be superior i
graph or network model [3, 4] pre



Informal Definitions

- Informally, a relation looks like a table of values.
- A relation contains a set of **rows**.
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
  - In the formal model, rows are called **tuples**
- Each column has a column header that gives an indication of the meaning of the data in that column
  - In the formal model, the column header is called an attribute name (or just **attribute**)

Example of a *INSTRUCTOR* Relation

INSTRUCTOR

attributes (or columns)

tuples (or rows)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Informal Definitions

- **Key** of a Relation:
  - Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
    - Called the **key**
  - In the INSTRUCTOR table, ID is the key
  - Sometimes row-ids or sequential numbers are assigned as keys to identify the rows
    - Called artificial key or surrogate key

Formal Definitions - Schema

- The schema (or description) of a relation:
  - Denoted by R(A1, A2, ...An)
  - R is the name of the relation
  - The attributes of the relation are A1, A2, ..., An
  - The degree (or arity) of a relation: no. of attributes (n)
  - A relation instance *r* defined over schema *R* is denoted by *r*(*R*).
- Example: CUSTOMER(Cid, Cname, Address, Phone#)
  - CUSTOMER is the relation name
  - Defined over the 4 attributes: Cid, Cname, Address, Phone#
- Each attribute has a domain (a set of valid values)
  - For example, the domain of Cid is 6 digit numbers.
  - Denoted as dom(A1), dom(A2), ...

Formal Definitions - Tuple

- An n-tuple (row) is an ordered list of n values (enclosed in angled brackets $< v_1, v_2, ..., v_n >$)
- Each value v_i, $1 \le i \le n$, is an element of dom(A_i) or is a special NULL value (discussed later)
- Attribute values are atomic; that is, indivisible.
- A row in the CUSTOMER relation is a 4-tuple consisting of 4 values, for example:
  – <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
- A relation r = $\{t_1, t_2, ..., t_m\}$ is a set of n-tuples

Formal Definitions - Domain

- A **domain** has a logical definition:
  – Example: "USA_phone_numbers" are the set of 10 digit phone numbers valid in the U.S.
- A domain also has a data-type or a format defined for it.
  – The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
  – Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.

Attribute Name and Domain

- The attribute name designates the role played by a domain in a relation:
  - Used to interpret the meaning of the data elements corresponding to that attribute
  - Example: The domain Date may be used to define two attributes “Invoice-date” and “Payment-date”
- More example: attribute Cname is defined over the domain of char strings of max length 25
  - dom(Cname) is varchar(25)
- The role these strings play in the CUSTOMER relation is that of the *name of a customer*.

Relation State

- The relation state is a subset of the Cartesian product of the domains of its attributes
- Each domain contains the set of all possible values the attribute can take.
- Cartesian product of the domains is the set of all possible combinations of attribute values (example in next slide)

Relation State - Examples

- Let $R(A_1, A_2)$ be a relation schema:
  - Let $dom(A_1) = \{0,1\}$
  - Let $dom(A_2) = \{a,b,c\}$
- Cartesian product of the domains: $dom(A_1) \times dom(A_2)$ is all possible combinations:

  {<0,a> , <0,b> , <0,c>, <1,a>, <1,b>, <1,c> }
- The relation state $r(R) \subset dom(A_1) \times dom(A_2)$
- Eg.: r(R) could be {<0,a> , <0,b> , <1,c> }
  - this is one possible **state** (or "population" or "extension") r of the relation R, defined over A_1 and A_2.
  - It has three 2-tuples: <0,a> , <0,b> , <1,c>

Formal Definitions - Summary

- Formally,
  - Given $R(A_1, A_2, ..., A_n)$
  - $r(R) \subset dom(A_1) \times dom(A_2) \times ... \times dom(A_n)$
- $R(A_1, A_2, ..., A_n)$ is the schema of the relation
- R is the name of the relation
- $A_1, A_2, ..., A_n$ are the attributes of the relation
- r(R): a specific state (or "value" or "population") of relation R – this is a *set of tuples* (rows)
  - $r(R) = \{t_1, t_2, ..., t_m\}$ where each t_i is an n-tuple
  - ti = $< v_1, v_2, ..., v_n >$ where each v_j is an *element-of* $dom(A_j)$

Definition Summary

Informal Terms		Formal Terms
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Characteristics Of Relations

- Ordering of tuples in a relation r(R):
  - The tuples are NOT ordered, even though they appear to be in the tabular form.
- Ordering of attributes in a relation schema R (and of values within each tuple):
  - Attributes in R(A_1, A_2, ..., A_n) and the values in t=< v_1, v_2, ..., v_n > are ordered.

More Example: STUDENT Relation

Relation Name → STUDENT

Attributes → (Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)

Tuples → (each row)

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Same state as previous Figure (with diff ordering of tuples)

Figure 5.2
The relation STUDENT from Figure 5.1 with a different order of tuples.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21

Characteristics of Relations

- Values in tuples:
  - All values are considered atomic (indivisible).
  - Each value in a tuple must be from the domain of the attribute for that column
    - If tuple t = < v_1, v_2, ..., v_n > is a tuple (row) in the relation state r of R(A_1, A_2, ..., A_n)
    - Then each v_i must be a value from *dom(A_i)*
  - Flat relational model
    - Composite and multivalued attributes not allowed
    - **First normal form** assumption

NULL

- NULL values
  - A special **NULL** value is used to represent values that are unknown or inapplicable.
- Meanings for NULL values:
  - Value unknown
  - Value exists but is not available
  - Attribute does not apply to this tuple (also known as value undefined)
- The null value causes complications in the definition of many operations.

Characteristics of Relations

- Interpretation (meaning) of a relation
  - **Assertion**
    - Each tuple in the relation is a **fact** or a particular instance of the assertion
  - **Predicate**
    - Values in each tuple interpreted as values that satisfy predicate

Relational Model Notation

- Relation schema R of degree n
  - Denoted by $R(A_1, A_2, ..., A_n)$
- Uppercase letters Q, R, S
  - Denote relation names
- Lowercase letters q, r, s
  - Denote relation states
- Letters t, u, v
  - Denote tuples

Relational Model Notation

- We refer to component values of a tuple t by:
  - $t[A_i]$ or $t.A_i$
  - This is the value v_i of attribute A_i for tuple t
- Similarly, $t[A_u, A_v, ..., A_w]$ and $t(A_u, A_v, ..., A_w)$ refers to the subtuple of t containing the values of attributes $A_u, A_v, ..., A_w$, respectively in t

Relational Model Constraints

- Constraints
  - Restrictions on the actual values in a database state
  - Derived from the rules in the miniworld that the database represents
  - Must hold on all valid relation states.
  - Three categories (below)
- Inherent model-based constraints or implicit constraints
  - Inherent in the data model

Relational Model Constraints

- Schema-based constraints or explicit constraints
  - Can be directly expressed in schemas of the data model
- Application-based or semantic constraints or business rules
  - Cannot be directly expressed in schemas
  - Expressed and enforced by application program

Domain Constraints

- An implicit constraint is the domain constraint
  - Every value in a tuple must be from the *domain of its attribute* (or **null**, if allowed for that attribute)
- Typically include:
  - Numeric data types for integers and real numbers
  - Characters
  - Booleans
  - Fixed-length strings
  - Variable-length strings
  - Date, time, timestamp
  - Money
  - Other special data types

Key Constraints

- No two tuples can have the same combination of values for all their attributes. (no duplicate tuples)
- Superkey of R is a set of attributes SK of R such that:
  - No two tuples in any valid state r(R) will have the same value for SK (can uniquely identify tuples)
  - That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK]
  - This condition must hold in *any valid state* r(R)
- Key (candidate 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 (continued)

- 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. (why?)
- 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

Examples of Keys

- **Example**: Consider the STUDENTS schema
  STUDENTS(SSN, StuID, Name, Major, Bdate)
- What are the candidate keys of STUDENTS?
- What is the primary key?
- Is (Name, Major) a superkey?
- What about (SSN, Name)?
- Is (SSN, Name) a candidate key? Why(not)?

Key Constraints (continued)

- A relation can have **several** candidate keys
- Primary key of the relation
  - A designated key among candidate keys
  - Underline attribute
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We choose SerialNo as the primary key
- Other candidate keys are also known as unique keys

Key Constraints (continued)

- The primary key value is used to uniquely identify each tuple in a relation
  - Provides the tuple identity
- Also used to reference the tuple from another relation
  - General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
  - Not always applicable – choice is sometimes subjective

CAR table with two candidate keys – LicenseNumber chosen as Primary Key

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 5.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

Database Schema & State

- Relational Database Schema:
  - A set **S** = {R_1, R_2, ..., R_n} of relation schemas that belong to the same database.
  - A set of integrity constraints **IC**
- Following slide shows a COMPANY database schema with 6 relation schemas
- Relational database state
  - Set of relation states ***DB*** = {r_1, r_2, ..., r_m}
  - Each r_i is a state of R_i and such that the r_i relation states satisfy integrity constraints specified in IC

COMPANY Database Schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Relational Database State

- Valid state
  - Satisfies all the constraints in the defined set of integrity constraints IC
- Invalid state
  - Does not obey all the integrity constraints

Populated Database State

- Each relation has tuples in its current relation state
- The relational database state is a union of all the individual relation states
- Whenever the database is changed, a new state arises
- Basic operations for changing the database:
  - INSERT a new tuple in a relation
  - DELETE an existing tuple from a relation
  - MODIFY an attribute of an existing tuple
- Next slide shows an example state for the COMPANY database

Populated Database State for COMPANY

Figure 5.6
One possible database state for the COMPANY relational database schema.

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

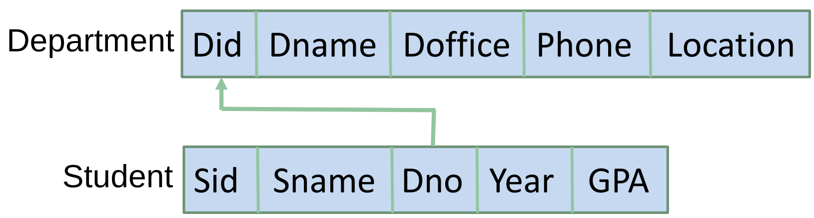
Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Entity Integrity

- The primary key attributes PK of each relation schema R in S cannot have null values in any tuple of r(R).
  - This is because primary key values are used to *identify* the individual tuples.
  - $t[PK] \neq null$ for any tuple t in r(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

- A constraint involving **two** relations
  - The previous constraints involve a single relation.
- Used to specify a **relationship** among tuples in two relations:
  - The referencing relation and the referenced relation.
  - Maintains consistency among tuples in two relations



Referential Integrity

- Tuples in the referencing relation R1 have attributes FK (called foreign key) that reference the primary key attributes PK of the referenced relation R2.
  - Value of FK in a tuple *t*1 of the current state *r*1*(R*1*)* either occurs as a value of PK for some tuple *t*2 in the current state *r*2*(R*2*)* or is NULL.
  - t1 is said to **reference** t2 if t1 [FK] = t2 [PK].
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Referential Integrity (or foreign key) Constraint

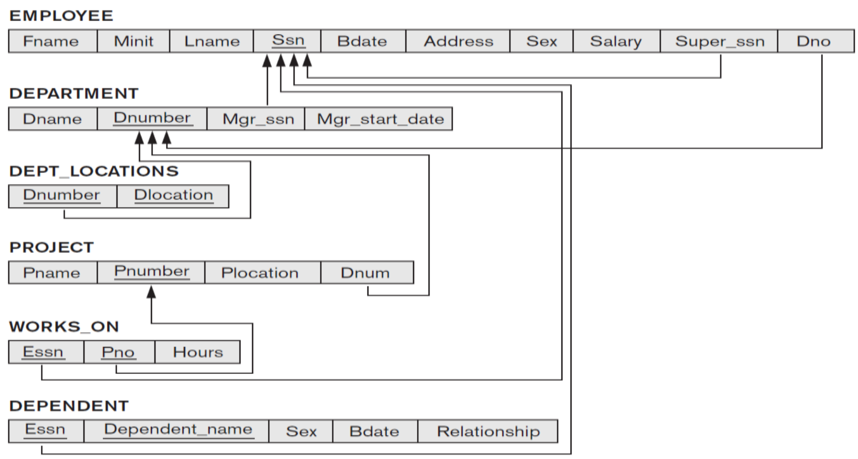
- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the **referencing relation** R1 can be **either**:
    - (1) a value of an **existing** primary key value of a corresponding primary key PK in the **referenced relation** R2, or
    - (2) a **null**.
- In case (2), the FK in R1 should **not** be a part of its own primary key.

Displaying a Schema and Constraints

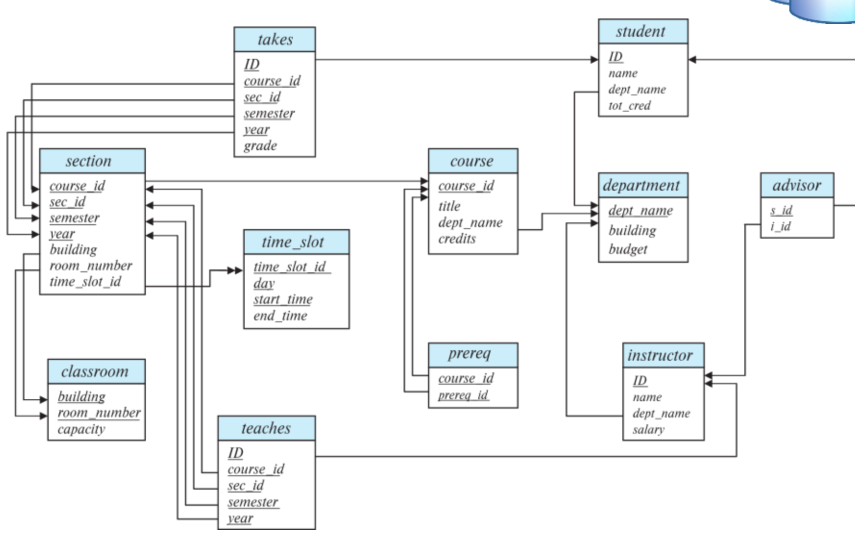
- Each relation schema can be displayed as a row of attribute names
- The name of the relation is written above the attribute names
- The primary key attribute (or attributes) will be underlined
- A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table
  - Can also point the the primary key of the referenced relation for clarity
- Next slide shows the COMPANY relational schema diagram

Relational Schema Diagram for COMPANY database

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Another Diagram Style



Other Types of Constraints

- Semantic integrity constraints
  - based on application semantics and cannot be expressed by the model per se
  - Example: “the max. no. of hours per employee for all projects is 56 hrs per week”
- A constraint specification language may have to be used to express these
- SQL allows triggers and assertions to express for some of these
  - More common to check for these types of constraints within the application programs

Other Types of Constraints

- Functional dependency constraint
  - Establishes a functional relationship among two sets of attributes *X* and *Y*
  - Value of *X* determines a unique value of *Y*
- State constraints
  - Define the constraints that a valid state of the database must satisfy
- Transition constraints
  - Define to deal with state changes in the database

Specification of a Relational Schema

- Select the relations, with a name for each table
- Select attributes for each relation and give the domain for each attribute
- Specify the key(s) for each relation
- Specify all appropriate foreign keys and integrity constraints
- Database schema is the set of schemas for the relations in a design

Update Operations on Relations

- Operations of the relational model can be categorized into retrievals and updates
- Basic operations that change the states of relations in the database:
  - Insert, Delete, Update (or Modify)
- Integrity constraints should not be violated by the update operations.
- 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.

Update Operations on Relations

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (RESTRICT or REJECT option)
  - Perform the operation but inform the user of the violation
  - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
  - Execute a user-specified error-correction routine

The Insert Operation

- Provides a list of attribute values for a new tuple t that is to be inserted into a relation R
- 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

The Delete Operation

- DELETE may violate only referential integrity:
  - If the primary key value of the tuple being deleted is referenced from other tuples
    - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL or SET DEFAULT (will discuss)
      - **RESTRICT** option: reject the deletion
      - **CASCADE** option: propagate the new primary key value into the foreign keys of the referencing tuples
      - **SET** option: set the foreign keys of the referencing tuples to NULL or default value
  - One of the above options must be specified during database design for each foreign key constraint

The Update Operation

- Need to specify a condition on attributes of relation
  - Select the tuple (or tuples) to be modified
- UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified
- Any of the other constraints may also be violated, depending on the attribute being updated:
  - Updating the primary key (PK):
    - Similar to a DELETE followed by an INSERT
    - Need to specify similar options to DELETE
  - Updating a foreign key (FK):
    - May violate referential integrity
  - Updating an ordinary attribute (neither PK nor FK):
    - Can only violate domain constraints

The Transaction Concept

- Transaction
  - Executing a designated function
  - Includes a sequence of operations
  - Considered as a single composite operation
  - Must leave the database in a valid or consistent state
- Online transaction processing (**OLTP**) systems
  - Execute transactions at rates that reach several hundred per second

Summary

- Relational model concepts
  - Definitions (informal and formal)
  - Characteristics of relations
- Relational model constraints and database schemas
  - Inherent model-based constraints, explicit schema-based constraints, and application-based constraints
  - Domain constraints, Key constraints, Entity integrity, Referential integrity
- Relational update operations and dealing with constraint violations

Exercise

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(SSN, Name, Major, Bdate)

COURSE(Course#, Cname, Dept)

ENROLL(SSN, Course#, Quarter, Grade)

BOOK_ADOPTION(Course#, Quarter, Book_ISBN)

TEXT(Book_ISBN, Book_Title, Publisher, Author)

Draw a relational schema diagram specifying the foreign keys for this schema.