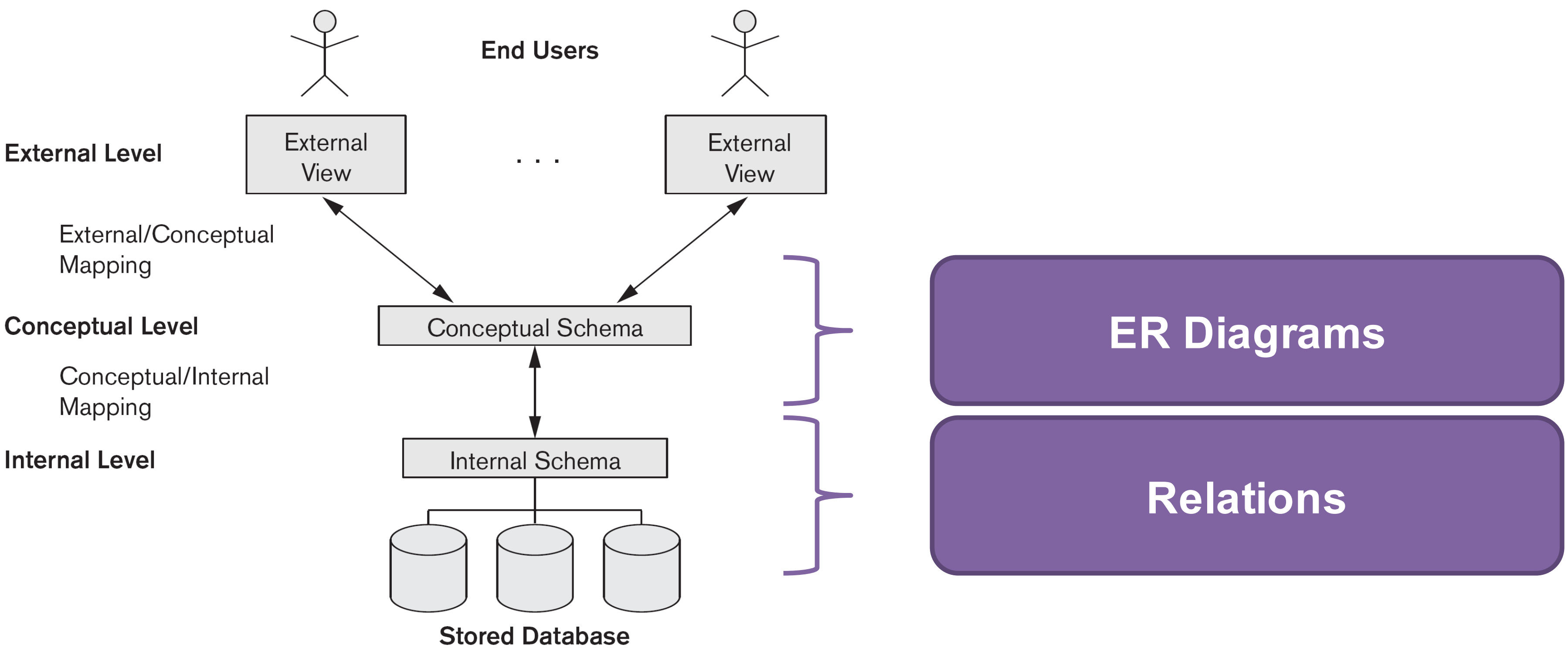
The Relational Data Model

(ALL the Vocabulary)

Lecture 2

A Quick Reminder

- One of the key features of a DBMS was to support "data independence" via data model(s)
  - The conceptual representation would be independent of underlying storage or operation implementation



Outline

1. Model Concepts
2. Model Constraints
3. Data Modification and Constraint Violation
4. Transactions

The Relational Model

Codd, Edgar F. "A relational model of data for large shared data banks." *Communications of the ACM* 13.6 (1970): 377-387.

"Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation)… 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…"

Information Retrieval P. BAXENDALE, Editor

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 models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity

CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

Volume 13 / Number 6 / June, 1970 Communications of the ACM 377

Motivation

- A **formal** mathematical basis for databases
  - Set theory and first-order predicate logic
  - Allows scientists to advance theoretically
- A foundation for efficient and usable database management systems
  - Allows companies/developers to advance end-user products
- Note: some aspects of the model are not adhered to by modern RDBMs

Relational Database

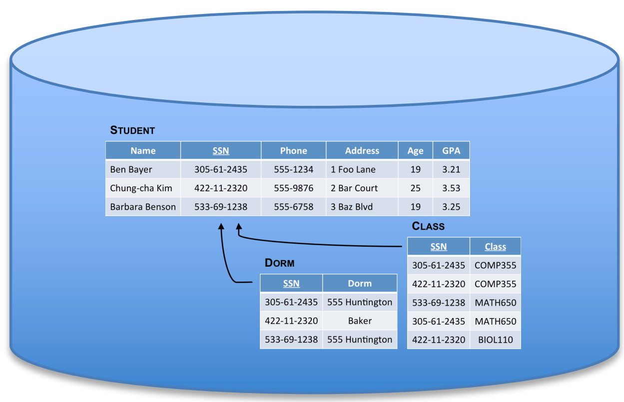
A database consists of…

i. a set of ***relations*** (tables)

ii. a set of ***integrity constraints***

Pop Quiz:
What is a **set**?

A database is in a **valid state** if it satisfies all integrity constraints (else **invalid state**)



STUDENT

Name	SSN	Phone	Address	Age	GPA
Ben Bayer	305-61-2435	555-1234	1 Foo Lane	19	3.21
Chung-cha Kim	422-11-2320	555-9876	2 Bar Court	25	3.53
Barbara Benson	533-69-1238	555-6758	3 Baz Blvd	19	3.25

DORM

SSN	Dorm
305-61-2435	555 Huntington
422-11-2320	Baker
533-69-1238	555 Huntington

CLASS

SSN	Class
305-61-2435	COMP355
422-11-2320	COMP355
533-69-1238	MATH650
305-61-2435	MATH650
422-11-2320	BIOL110

A Relation

A relation consists of…

i. its ***schema***, describing structure
ii. its ***state***, or current populated data

Schema: STUDENT

Name	SSN	Phone	Address	Age	GPA
Ben Bayer	305-61-2435	555-1234	1 Foo Lane	19	3.21
Chung-cha Kim	422-11-2320	555-9876	2 Bar Court	25	3.53
Barbara Benson	533-69-1238	555-6758	3 Baz Blvd	19	3.25

State: (the rows above)

Relational Schema

- Relation name
  STUDENT

- Ordered list of *n* ***attributes*** (columns; degree *n* or *n*-ary)
  Each with a corresponding ***domain*** (list of valid ***atomic*** values)
  – dom(SSN) = "###-##-####"
  – dom(GPA) = [0, 4]

- Notation: NAME(A_1, A_2, … A_n)
  STUDENT(Name, SSN, Phone, Address, Age, GPA)

What is the degree of STUDENT?

STUDENT

Name	SSN	Phone	Address	Age	GPA

Relation State

- A set of *n*-**tuples** (rows)
  - Each has a value in the domain of every corresponding attribute (or **NULL**)
  - Notation: *r*(NAME)
- Mathematically, a subset of the Cartesian product of the attribute domains; related to the closed-world assumption

$$r(STUDENT) \subseteq (dom(Name) \times dom(SSN) \times \ldots dom(GPA))$$

Ben Bayer	305-61-2435	555-1234	1 Foo Lane	19	3.21
Chung-cha Kim	422-11-2320	555-9876	2 Bar Court	25	3.53
Barbara Benson	533-69-1238	555-6758	3 Baz Blvd	19	3.25

Exercise

Diagrammatically produce a relation HAT according to the following schema; the relation state should have at least three tuples

HAT(Team, Size, Color)

- dom(Team) = { `RedSox, Bruins, Celtics, Patriots, Revolution` }
- dom(Size) = { `S, M, L, XL` }
- dom(Color) = { `Black, Blue, White, Red, Green, Yellow` }

How many tuples are possible in this relation?

Answer

HAT

Team	Size	Color
RedSox	M	Red
Revolution	S	White
Bruins	XL	Yellow

$$|dom(Team)| \times |dom(Size)| \times |dom(Color)|$$

$$5 \times 4 \times 6$$

$$120$$

Tuples: Theory vs. Implementation

- Relation state is formally defined as a *set* of tuples, implying…
  - No inherent order
  - No duplicates
- In real database systems, the rows on disk will have an ordering, but the relation definition sets no preference as to this ordering
  - We will discuss later in physical design how to establish an ordering to improve query efficiency
- Additionally, real database systems implement a *bag* of tuples, allowing duplicate rows

NULL

- **NULL** is a special value that may be in the attribute domain
- Several possible meanings
  - E.g. unknown, not available, does not apply, undefined, …
- Best to avoid
  - Else deal with caution

Value Structure in Tuples

- Each value should be **atomic** – no *composite* or *multi-valued* attributes
  - Composite: "one column, many parts"
  - Multi-valued: "one column, multiple values"
- Convention called 1NF (*first normal form*)
  - More on this later in the course

Violation of 1NF: Composite

DORM

	Dorm Info	
<u>SSN</u>	Dorm	Room
305-61-2435	555 Huntington	1
422-11-2320	Baker	2
533-69-1238	555 Huntington	3

vs.

DORM

<u>SSN</u>	Dorm	Room
305-61-2435	555 Huntington	1
422-11-2320	Baker	2
533-69-1238	555 Huntington	3

Violation of 1NF: Multi-Valued

CLASS

SSN	Class
305-61-2435	COMP355, MATH 650
422-11-2320	COMP355, BIOL110
533-69-1238	MATH650

vs.

CLASS

SSN	Class
305-61-2435	COMP355
422-11-2320	COMP355
533-69-1238	MATH650
305-61-2435	MATH650
422-11-2320	BIOL110

Model Constraints

Categories of restrictions on data in a relational database

1. Inherent in the data model (implicit)
2. Schema-based (explicit)
3. Application-based (or triggers/assertions)
4. Data dependencies

   Relates to "goodness" of database design; we will revisit in normalization

Schema-Based Constraints

Can be directly expressed in schemas of the data model, typically by specifying them in the **DDL** (Data Definition Language)

- Domain
- Key
- Entity integrity
- Referential integrity

Domain Constraints

Within each tuple, the value of each attribute A must be an atomic value from the domain dom(A)

Schema must dictate whether or not a NULL value is allowed for each attribute

$$NULL \overset{?}{\in} dom(A)$$

More later on standard data types in SQL

Key Constraints

A **key** is a set of attribute(s) satisfying two properties:

1. Two distinct tuples in any state of the relation cannot have identical values for all the attributes of the key (**superkey**)

2. No attribute can be removed from the key and still have #1 hold (**minimal superkey**)

A relation may have multiple keys (each is a **candidate key**). Relations commonly have a **primary key** (underlined, PK; typically small number of attributes, used to *identify* tuples), and may also have some number of additional **unique key(s)**.

Exercise

Is the following a valid state of DOCTOR?

DOCTOR

Number	First	Last
1	William	Hartnell
2	Patrick	Troughton
3	Jon	Pertwee
4	Tom	Baker
5	Peter	Davison
6	Colin	Baker
7	Sylvester	McCoy
8	Paul	McGann
9	Christopher	Eccleston
10	David	Tennant
11	Matt	Smith
12	Peter	Capaldi

Answer

Is the following a ~~valid~~ state of DOCTOR?

Doctor

Number	First	Last
1	William	Hartnell
2	Patrick	Troughton
3	Jon	Pertwee
4	Tom	Baker
5	Peter	Davison
6	Colin	Baker
7	Sylvester	McCoy
8	Paul	McGann
9	Christopher	Eccleston
10	David	Tennant
11	Matt	Smith
12	Peter	Capaldi

Underline = **primary key** = First
Key requirement #1: Two distinct tuples in any state of the relation cannot have identical values for all the attributes of the key – **NOT TRUE!**

Exercise

List all keys for the current state of DOCTOR.

DOCTOR

Number	First	Last
1	William	Hartnell
2	Patrick	Troughton
3	Jon	Pertwee
4	Tom	Baker
5	Peter	Davison
6	Colin	Baker
7	Sylvester	McCoy
8	Paul	McGann
9	Christopher	Eccleston
10	David	Tennant
11	Matt	Smith
12	Peter	Capaldi

Answer

List <u>all</u> keys for the current state of DOCTOR.

DOCTOR

Number	First	Last
1	William	Hartnell
2	Patrick	Troughton
3	Jon	Pertwee
4	Tom	Baker
5	Peter	Davison
6	Colin	Baker
7	Sylvester	McCoy
8	Paul	McGann
9	Christopher	Eccleston
10	David	Tennant
11	Matt	Smith
12	Peter	Capaldi

Candidate Key #1: { Number }
Candidate Key #2: { First, Last }

Why not { Last }, { Number, Last }?

Entity Integrity

In a tuple, no attribute that is part of the PK can be NULL

Basic justification: if PK is used to identify a tuple, then none of its component parts can be left unknown

Exercise

List all potential primary keys for the current state of DOCTOR.

DOCTOR

Number	First	Last
1	William	Hartnell
2	Patrick	Troughton
3	Jon	Pertwee
4	Tom	Baker
5	Peter	Davison
6	Colin	Baker
7	Sylvester	McCoy
8	Paul	McGann
9	Christopher	Eccleston
10	David	Tennant
11	Matt	Smith
12	Peter	Capaldi
13	NULL	NULL

Answer

List all potential primary keys for the current state of DOCTOR.

DOCTOR

Number	First	Last
1	William	Hartnell
2	Patrick	Troughton
3	Jon	Pertwee
4	Tom	Baker
5	Peter	Davison
6	Colin	Baker
7	Sylvester	McCoy
8	Paul	McGann
9	Christopher	Eccleston
10	David	Tennant
11	Matt	Smith
12	Peter	Capaldi
13	NULL	NULL

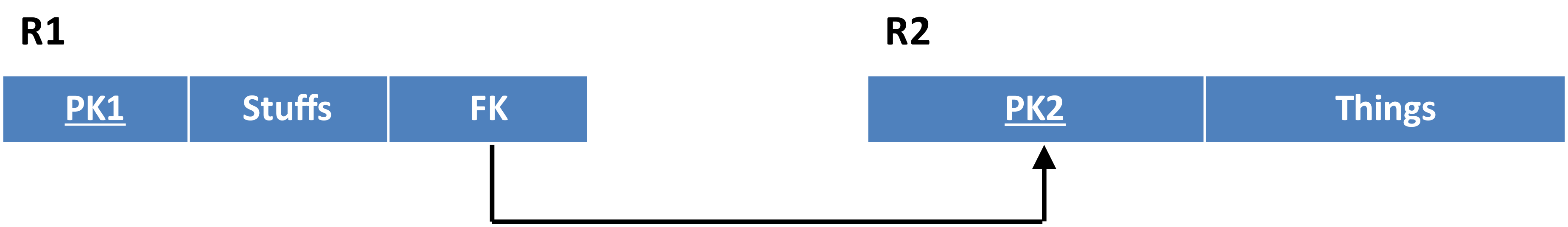
PK = { Number }

Referential Integrity

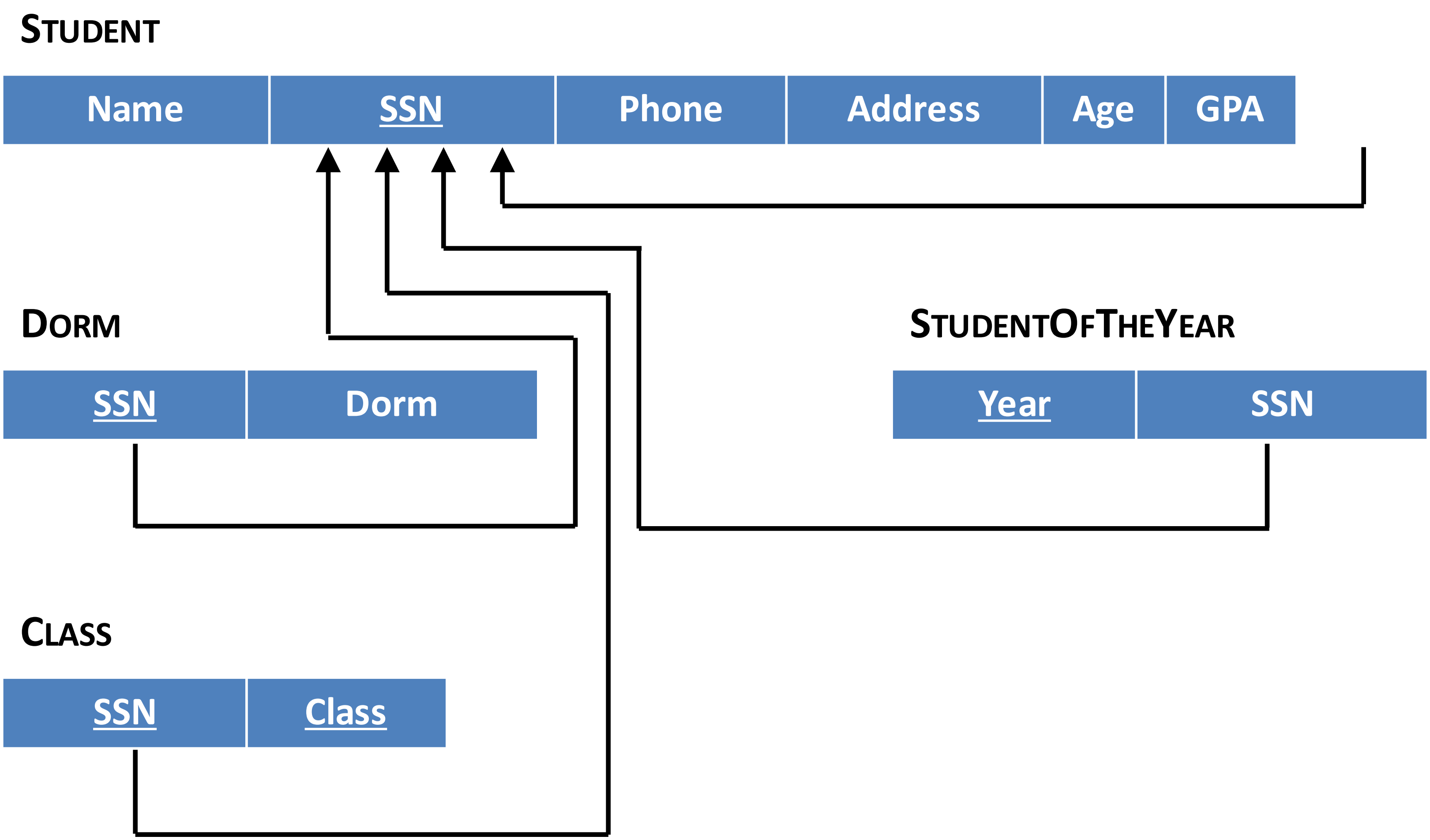
All tuples in relation R1 must reference an existing tuple in relation R2 (R1 *may* be the same as R2)

A **foreign key** (FK) in R1 references R2 iff…

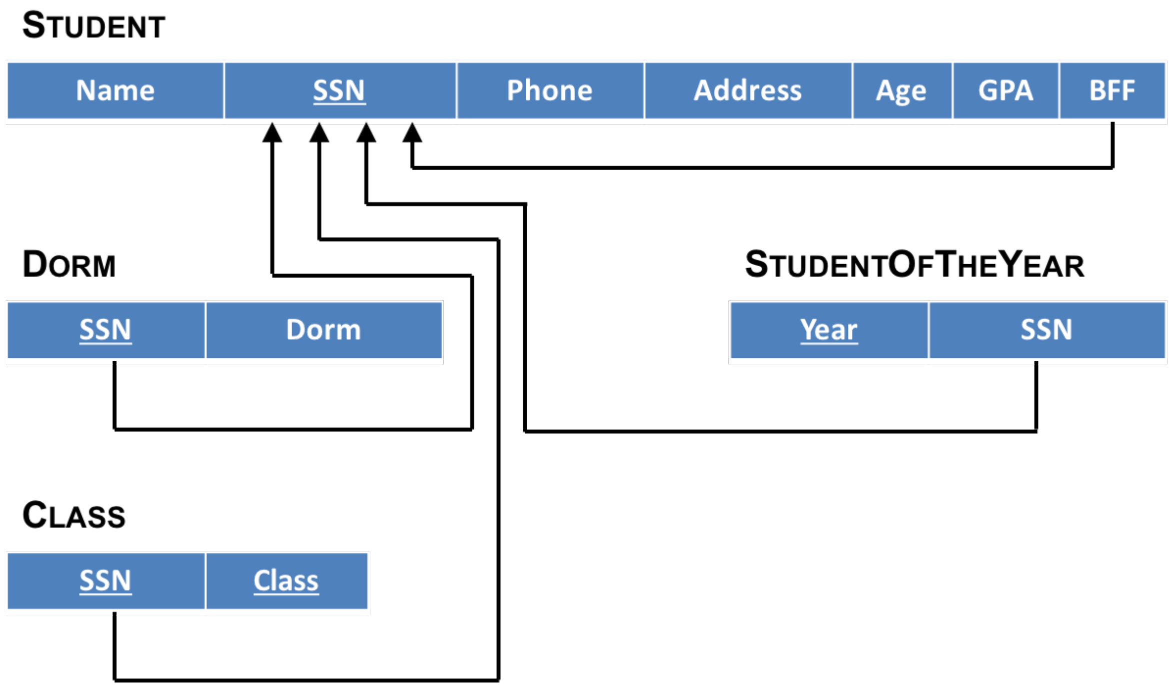
- The attribute(s) in FK have the same domain(s) as the primary key attribute(s) PK of R2
- A value of FK in a tuple t1 either is NULL or occurs as a value of PK for some tuple t2 (t1 *refers to* t2)



Example

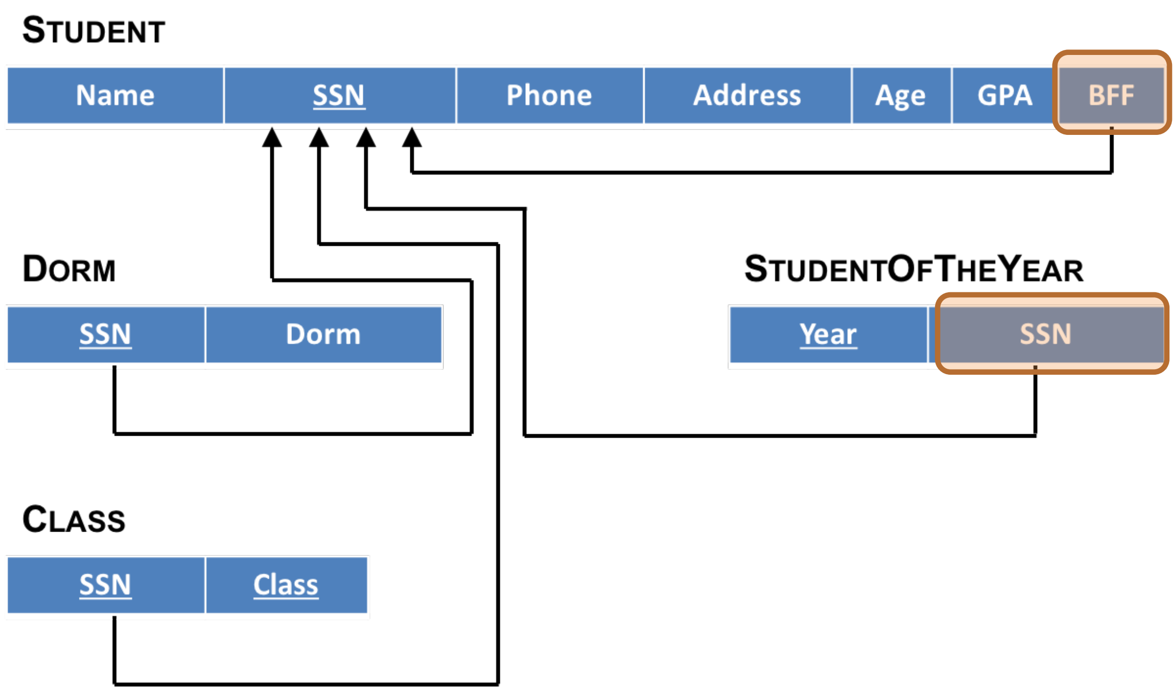

STUDENT
Name
SSN
Phone
Address
Age
GPA
DORM
SSN
Dorm
STUDENTOFTHEYEAR
Year
SSN
CLASS
SSN
Class

Exercise



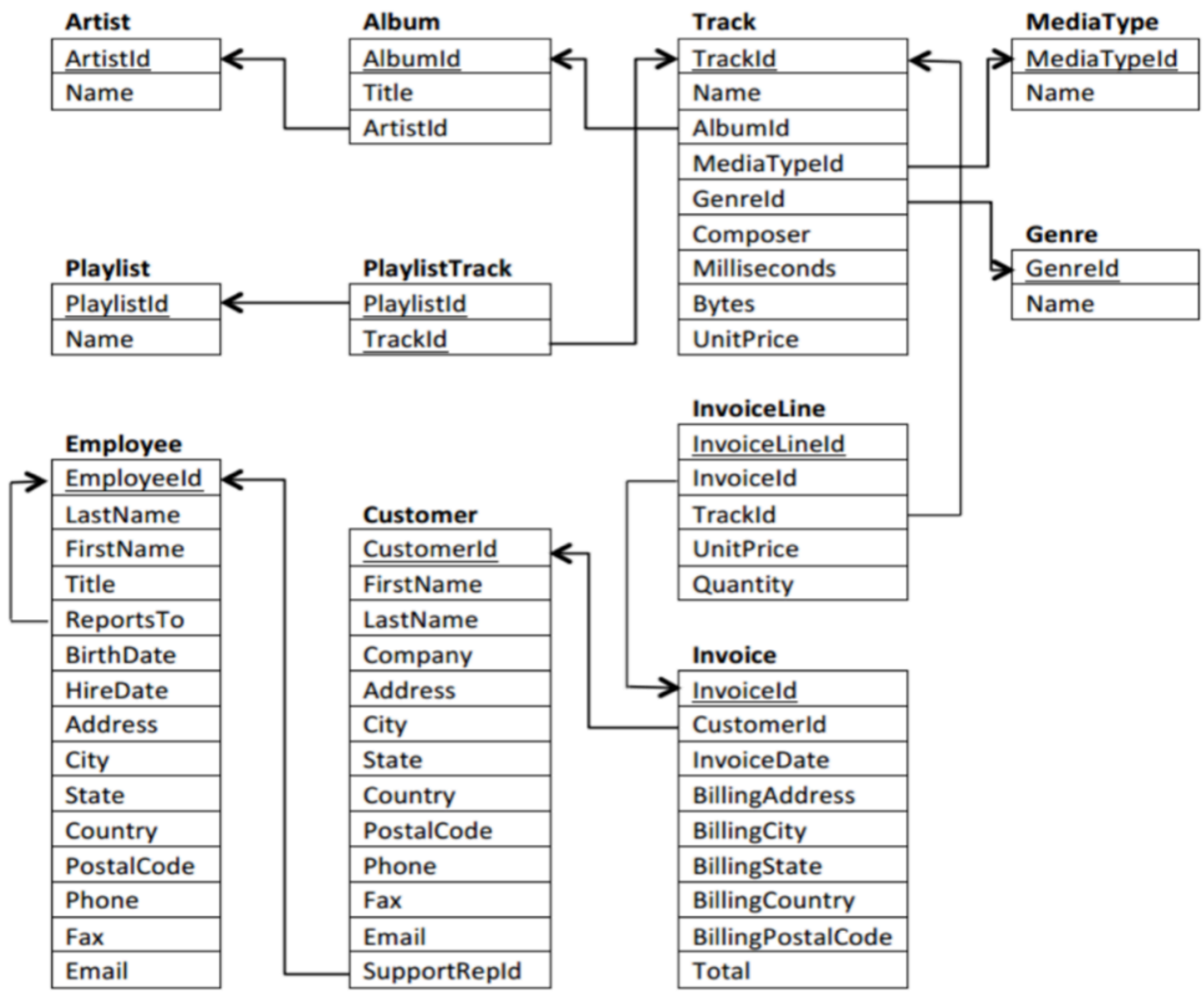
Given the above relational schema, for which attribute(s) that refer to STUDENT(SSN), if any, is it permissible to have a value of NULL?

Answer



Given the above relational schema, for which attribute(s) that refer to STUDENT(SSN), if any, is it permissible to have a value of NULL?

Chinook


Artist
ArtistId
Name
Album
AlbumId
Title
ArtistId
Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice
MediaType
MediaTypeId
Name
Genre
GenreId
Name
Playlist
PlaylistId
Name
PlaylistTrack
PlaylistId
TrackId
InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity
Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email
Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId
Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total

Data Modification Operations

The **DML** (Data Manipulation Language) affords us the following methods of modifying database state:

- **Insert**. Add a new tuple to a relation
- **Delete**. Remove a tuple from a relation
- **Update**. Change one or more attribute value(s) for a tuple within a relation

We now examine how these operations can violate various types of constraints and the resulting actions that can be taken

Insert

Domain

- An attribute value does not appear in the corresponding domain (including NULL)

Key

- A key value already exists in another tuple

Entity Integrity

- Any part of the primary key is NULL

Referential Integrity

- Any value of any foreign key refers to a tuple that does not exist in the referenced relation

Typical action: reject insertion

Delete

Referential Integrity

- Tuple being deleted is referenced by foreign keys from other tuples

Possible actions

- Reject deletion
- Cascade (propagate deletion)
- Set default/NULL referencing attribute values (careful with primary key)

Update

- If modifying neither part of primary key nor foreign key, need only check…
  - **Domain**

- Modifying primary key…
  - Like **Delete** then **Insert**

- Modifying foreign key…
  - Like **Insert**

Actions typically similar to Delete with separate options.

Transactions

A **transaction** is a sequence of database operations, including retrieval and update(s)

START

Read or write

Read or write

Read or write

…

COMMIT or **ROLLBACK**

Desirable Properties of Transactions

A **tomicity**. A transaction is an atomic unit of processing; it should either be performed in its entirety or not performed at all.

C **onsistency**. A transaction should be consistency preserving, meaning that if it is completely executed from beginning to end without interference from other transactions, it should take the database from one consistent state to another.

I **solation**. A transaction should appear as though it is being executed in isolation from other transactions, even though many transactions are executing concurrently. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.

D **urability**. The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.

Exercise

Classify each of the following statements with the best-matching property (ACID)

1. *For a balanced budget, incoming funds must always equal outgoing payments*
2. *Once a package is confirmed as received, it must be delivered*
3. *If there is an error in printing a picture at the photo booth, the customer should be refunded*
4. *Do not publish results while the jury is out*

Answer

1. *For a balanced budget, incoming funds must always equal outgoing payments*

   Consistency

2. *Once a package is confirmed as received, it must be delivered*

   Durability

3. *If there is an error in printing a picture at the photo booth, the customer should be refunded*

   Atomicity

4. *Do not publish results while the jury is out*

   Isolation

Summary

- The **relational model** dictates that a relational database consists of (i) a set of relations and (ii) a set of integrity constraints
  - All constraints met => database in a **valid** state
- A relation is composed of its **schema** (name; list of n attributes, each with its **domain**) and its **state**/data (set of n-**tuples**)
- Schema (or **explicit**) constraints, specified via **DDL**, include domain, key, entity integrity, and referential integrity
  - Data manipulation operations (insert, update, delete; via **DML**) can run awry of these constraints
- A **transaction** is a sequence of operations and **ACID**-compliant RDBMSs implement ”proper” transaction processing
  - **A**tomicity, **C**onsistency, **I**solation, **D**urability