COMP9311: Database Systems

Relational Data Model

(textbook: chapters 5)

Term 3 2022

Week 2 Relational Data Model

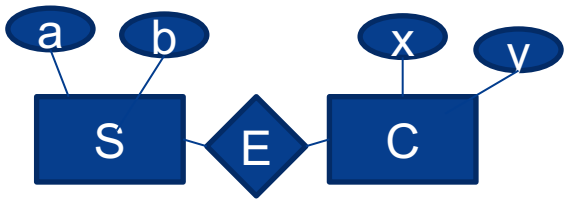
By Helen Paik, CSE UNSW

Disclaimer: the course materials are sourced from

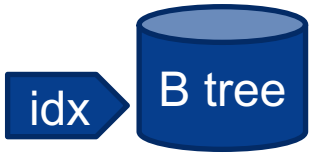
- previous offerings of COMP9311 and COMP3311
- Prof. Werner Nutt on Introduction to Database Systems (http://www.inf.unibz.it/~nutt/Teaching/IDBs1011/)

Relational Data Model

Data can be represented at different levels of abstraction



S (a,b,…)
C (x,y,…)



Conceptual Schema → Logical Schema → Physical Schema

ER:
- Entities,
- Relationships,
- Attributes

Not specifically tied to a database

Relations:
- tuples,
- attributes
- domain

≈ Tables/columns/rows

specifically tied to the data model of a database you chose (for our course, the choice is relational database)

File organisation:
- File types
- Index structures

specifically tied to the implementation
Methods provided by the database you chose

Relational Data Model Concepts

The relational data model is the most widely used data model for database systems.

The *relational data model* describes the world as

- a **collection** of inter-connected ***relations***

Goal of relational model:

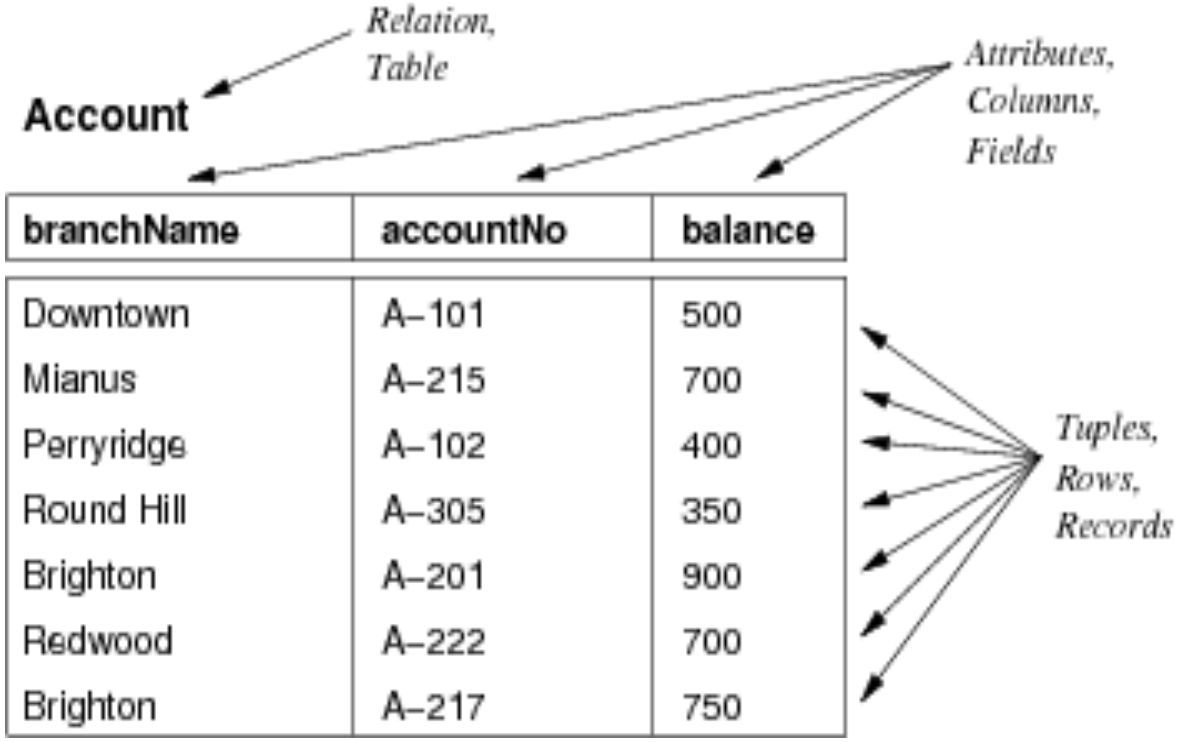
- a simple, general data modelling formalism
- which maps easily to file structures (i.e. implementable)

Relational model has **two styles** of terminology:

- mathematical: relation, tuple, attribute, ...
- data-oriented: table, record, field/column, ...

Warning: textbooks alternate between the two; treat them as synonyms

Example relation: bank accounts (terminology)



branchName	accountNo	balance
Downtown	A–101	500
Mianus	A–215	700
Perryridge	A–102	400
Round Hill	A–305	350
Brighton	A–201	900
Redwood	A–222	700
Brighton	A–217	750

- In a relation, each row (a.k.a. tuple) represents a collection of related data values and the names (i.e., table name and column names) help interpret the meaning of the values

Relational Data Model - formal definitions

The relational model has one *structuring mechanism* ...

- **a *relation* corresponds to a mathematical "relation"**

A relation schema (denoted *R,S,T,...*) has:

- a *name* (unique within a given database)
- a set of *attributes* (which can be viewed as column headings)
- e.g., STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)
- or generally, $R\ (A_1, A_2, \ldots, A_n)$

Each *attribute* (denoted *A,B,...* or $a_1,a_2,...$) has:

- a *name* (unique within a given relation)
- an associated *domain* (set of allowed values)

e.g., STUDENT(Name: *string*, Ssn: string, …, Age: integer, Gpa: real)

Relational Model - formal definitions

A relation schema R is formally denoted by:

$R(A_1, A_2, A_3 \ldots, A_n)$

where A_i denotes an attribute in R.

e.g., STUDENT(Name, Sid, Age, GPA)

Domain refers to the legal type and range of values for an attribute, denoted by $dom(A_i)$

- e.g., Attribute Age — Domain: [0-100]
- e.g., Attribute EmpName — Domain: 50 alphabetic chars
- e.g., Attribute Salary — Domain: non-negative integer
- Domain can also specify the format of the attribute (e.g., (ddd)dd—dddd))

Sometimes R can be written as:

$R(A_1:D_1, A_2:D_2, \ldots A_n:D_n)$

where A_i denotes an attribute in R, D_i denotes the domain of A_i

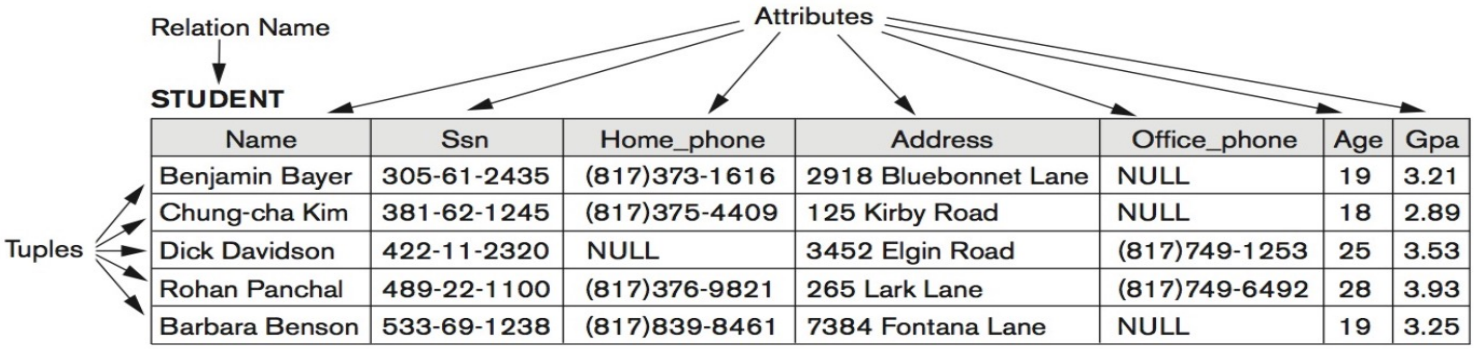
e.g., STUDENT(Name: string, Sid: string, Age: integer, GPA: real)

Relational Model – formal definitions

A relation r of the relation schema $R(A_1, A_2, \ldots, A_n)$ is denoted by $r(R)$

$r(R)$ is *a set of n-tuples*, i.e., $r = \{t_1, t_2, \ldots, t_m\}$

Each tuple t is *an ordered list* of values $t = <v_1, v_2, v_3 \ldots, v_n>$ where each v_i is an element of $dom(A_i)$



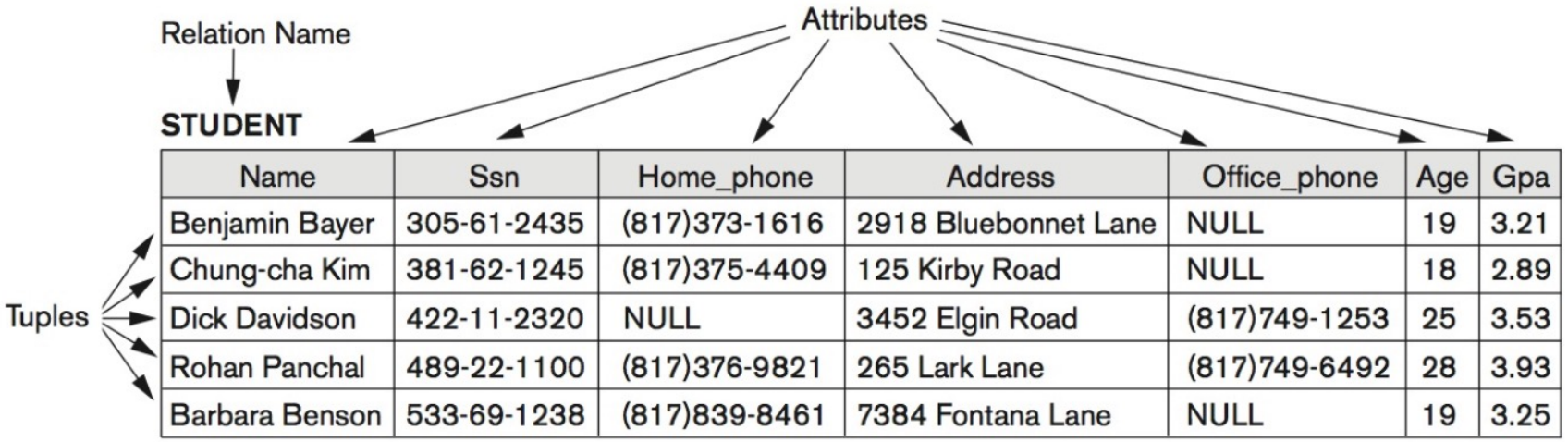
Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821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

$R_{student}$ (Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)

r_1 = {<Benjamin Bayer, 305-61-2435, (817)373-1616, …, 3.21>,
<Chung-cha Kim, 381-62-1245, …>, … <Barbara Benson, …>}

The i^{th} value in tuple t_j, corresponds to the attribute A_i, referred to as $t[A_i]$ or $t.A_i$ or $t[i]$)
e.g., Benjamin Bayer is $t_1[1]$, $t_1[Name]$, or $t_1.Name$

Relation Schema (R) vs Relation Instance r(R)



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

Analogy with programming languages:

schema = type

instance = value

Important distinction:

- Relation Schema = stable over long periods of time
- Relation Instance = **a relation state**, it changes constantly, as data is inserted/updated/deleted

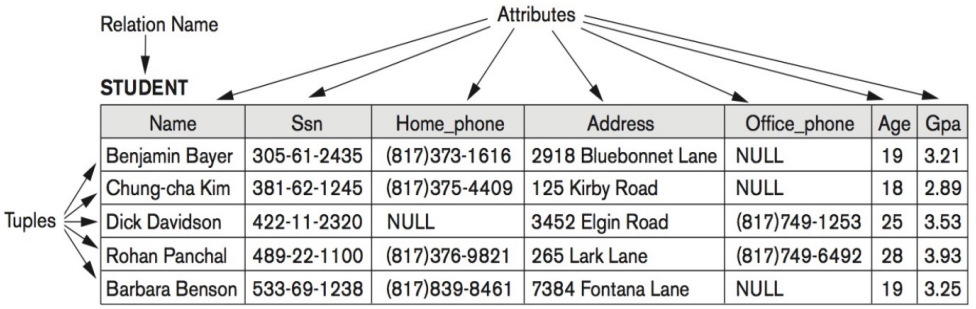
Characteristics of Relations

- **<u>Ordering of Tuples in a Relation:</u>** a relation is defined as a set of tuples → no order among them

- Tuples in a relation do not have any particular order (r_1, r_2 on the right are identical from the relation model view point)

- Tuple ordering is not part of a relation because the relation is only capturing the logical level schema.

- When the tuples are physically stored in a file (disk), an order may be enforced by the storage mechanism. But this concept is not part of a relation.



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

STUDENT

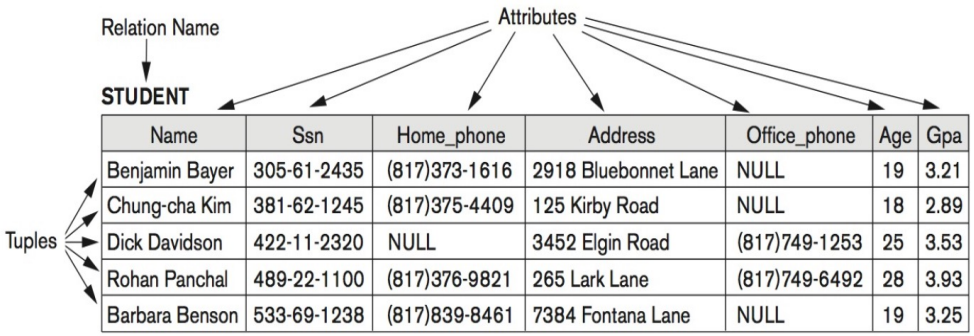
Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Chung-cha Kim	381-62-1245	(817)375-4409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

- **Ordering of Values within a Tuple**: a tuple is an order list of values → there is an order among them

- The order of attributes in a relational schema is therefore important …

- More abstract level relation schema can be defined so that the order of attributes do not matter. Refer to pp. 184-185. But we will mostly use the first definition.



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

$R_{student}$ (Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)

t_1 = <Benjamin Bayer, 305-61-2435, (817)373-1616, …, 3.21> (correct)

t_1 = <Benjamin Bayer, (817)373-1616, 305-61-2435, …, 3.21> (wrong)

Characteristics of Relations

- **Values and NULLs in the Tuples:** each value in a tuple is an atomic value
- e.g., “Benjamin Bayer” is a single string value (not two words)
- Person.Favourite_Foods = <“Thai, Indian”> is also a single string value (not two words).

- *A tuple value could be NULL* – which represent the values of attributes that may be unknown or does not apply to the tuple.

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

NULL values are sometimes ambiguous
- Unknown
- Not available
- Does not apply
- …

NULL values could be problematic when performing Some operations (like comparison).

Is Babara’s phone number the same as Benjamin’s phone number?
Both NULL, but not the same

Relational Model and Integrity Constraints

- In a relational database system, we are likely to have many relations – and the tuples in those relations are usually related in various ways (e.g., a student tuple in Student is enrolled in a course tuple in Course)
- **Ideal**: relation instances in the db system reflect the real world correctly
- **In real life**: This is not always the case (e.g., wrong tuples)
- **Goal**: Reduce such errors as much as we can
- **Observation**: Not all mathematically possible instances make sense
- **Idea**:
  - Formulate conditions that hold for all plausible instances
  - Check whether the condition holds when updating a relation
  - Such conditions are called *integrity constraints*

Common Types of Integrity Constraints

Constraints Applying to Single Relations

Domain Constraints

- "No employee is younger than 15 or older than 80"
- `dom(Age) = > 15 or < 80`

Superkeys and keys

- *"Employees with the same tax code are identical", in other words, the values of any given two employees' tax code attribute are different*
- $t_1[taxCode] \neq t_2[taxCode]$

NULL value constraint

- "Employee name cannot be NULL"

Common Types of Integrity Constraints

Constraints Applying to Many Relations

Example Relation Schema

Account	branchName	accountNo	balance	
Branch	branchName	address	assets	
HeldBy	account	customer		
Customer	name	Address	customerNo	homeBranch

Common Types of Integrity Constraints

Example Instances

Account

branchName	accountNo	balance
Downtown	A–101	500
Mianus	A–215	700
Perryridge	A–102	400
Round Hill	A–305	350
Brighton	A–201	900
Redwood	A–222	700

Branch

branchName	address	assets
Downtown	Brooklyn	9000000
Redwood	Palo Alto	2100000
Perryridge	Horseneck	1700000
Mianus	Horseneck	400000
Round Hill	Horseneck	8000000
North Town	Rye	3700000
Brighton	Brooklyn	7100000

Customer

name	address	customerNo	homeBranch
Smith	Rye	1234567	Mianus
Jones	Palo Alto	9876543	Redwood
Smith	Brooklyn	1313131	Downtown
Curry	Rye	1111111	Mianus

HeldBy

account	customer
A–101	1313131
A–215	1111111
A–102	1313131
A–305	1234567
A–201	9876543
A–222	1111111
A–102	1234567

Common Types of Integrity Constraints

Entity Integrity Constraint

- No primary key value can be NULL
- Applies to a single relation, but important in the context of multiple relations

Referential Integrity Constraint

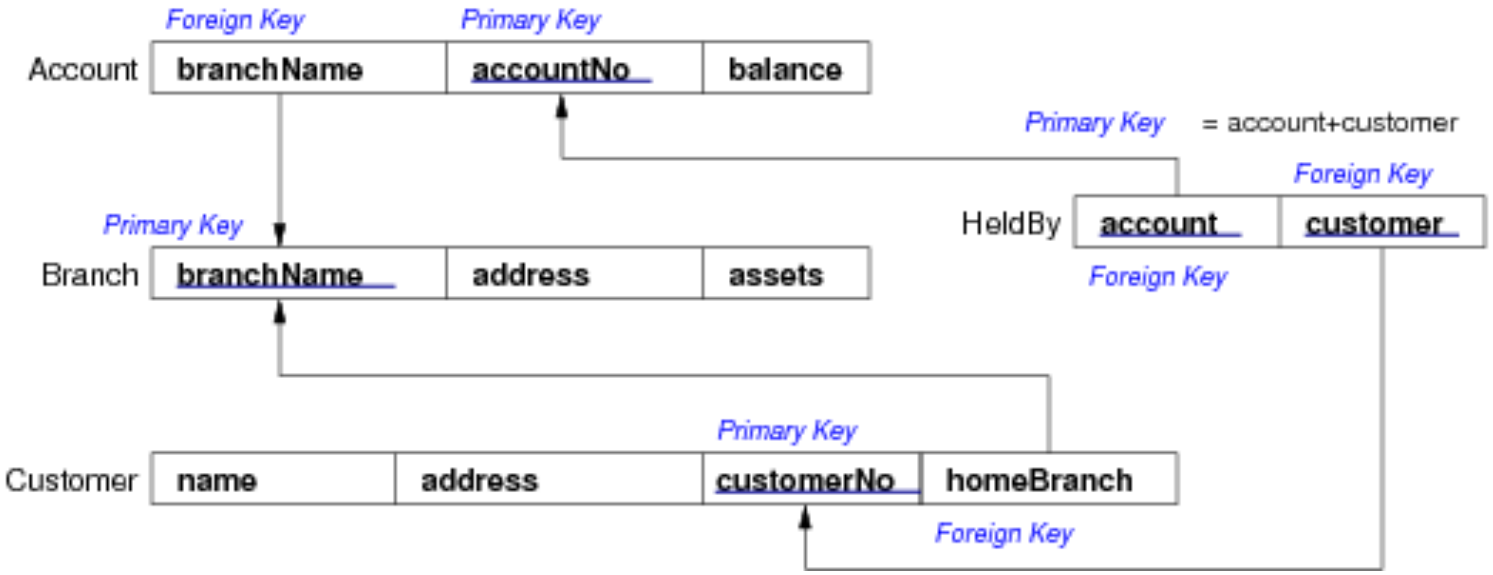
- Between two relations, a tuple in one relation that refer to another relation must refer to an existing tuple in that relation
- For example, Account.branchName must refer to an existing tuple (i.e., branch name) in Branch.branchName
- Or HeldBy.customer must refer to an existing tuple (i.e., customer number) in Customer.customerNo

- (note: NULL or not NULL)

Referential Integrity

Referential integrity constraints

- describe references between relations (tables)
- are related to notion of a *foreign key* (FK), i.e., a key from another relation



Notation:

FK1: Account (branchName) references Branch (branchName)

FK2: HeldBy (account) references Account (accountNo)

- Attribute names could be different, but the domains must be the same

Referential Integrity

More formally describing a foreign key here …

A set of attributes *F* in relation R_1 is a *foreign key* from R_2 if:

- the attributes in *F* correspond to the primary key of R_2
- the value for *F* in each tuple of R_1
  - either occurs as a primary key in R_2 (i.e., existing value)
  - or is entirely NULL*

R_1 Account	branchName	accountNo	balance

R_2 Branch	branchName	address	assets

Foreign keys are critical in relational DBs; they provide ...

- the "glue" that links individual relations (tables)
- the way to assemble queries from multiple tables (e.g., list all accounts per branch)
- the relational representation of ER relationships

Updates and Dealing with Constraint Violations

A database reflects the state of an aspect of the real world:

The world changes ➔ the database has to change

Updates to an instance:

- adding a tuple
- deleting a tuple
- modifying an attribute value of a tuple
- updates to the data happen very frequently.

Updates to a schema:

- What could be updates to a schema?
  - e.g., deleting an attribute, changing the domain of an attribute, etc.
- Updates to the schema: relatively rare, rather painful, sometimes not possible.

Database updates may violate constraints …

Let's consider updates:

"Insertions, Deletions, Modifications" of tuples

Example DB with tables:

Student (**studno**, name, hons, tutor, year)

Staff (**lecturer**, roomno, appraiser)

FK: Student(tutor) references Staff(lecturer)

Assume that this database has "key" and "foreign key (i.e., referential integrity)" constraints … what can go wrong? How should the DBMS react?

Insertions (1)

If the following tuple is inserted into Student, what should happen? Why?

<s1, jones, cis, capon, 3>

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Insertions (2)

If the following tuple is inserted into Student, what should happen? Why?

<null, jones, cis, capon, 3>

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Insertions (3)

If the following tuple is inserted into Student, what should happen? Why?

<s7, jones, cis, null, 3>

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Insertions (4)

If the following tuple is inserted into Student, what should happen? Why?

<s7, jones, cis, calvanese, 3>

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Deletions (1)

If the following tuple is deleted from Student, is there a problem? And what should happen?

<s2, brown, cis, kahn, 2>

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Deletions (2)

And if this one is deleted from Staff ?

<kahn, IT206, watson>

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Modifications (1)

What if we change in Student

<s1, jones, ca, bush, 2>

to

<s1, jones, ca, watson, 2> ?

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Modifications (2)

And what if we change in Student

<s2, brown, cis, kahn, 2>

to

<s1, jones, ca, bloggs, 2> ?

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Modifications (3)

And what if we change in Staff

<lindsey, 2.10, woods>

to

<lindsay, 2.10, woods> ?

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Modifications (4)

Now, let's change in Staff

<goble, 2.82, capon>

to

<gobel, 2.82, capon> …

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Summary: Integrity Violations of Database Operations

The Insert Operation

- Domain constraints (e.g., a value out side of the defined domain)
- Key constraints (e.g., the value that already exists in another tuple)
- Entity integrity (e.g., if the value of the primary key is NULL)
- Referential integrity (e.g., the value of any FK does not exist in the referenced relation)
- *Reject the operation*

The Delete operation

- The tuple being deleted is referenced by FKs from other tuples in other relations
- *Reject or "cascade delete", or "set to NULL"*

The Update operation

- Modifying primary key or FK may have the same effect/considerations as DELETE
- Updating non-key attributes are usually OK (maybe domain constraints)