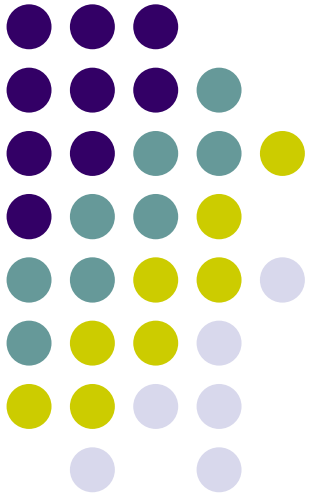
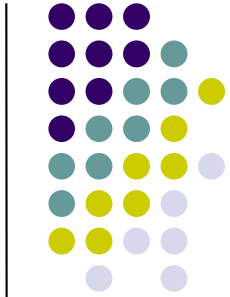
ER to Relation



Review: Relational Data Model

Key Abstraction: Relation



Mathematical relations

Given sets: R = {1, 2, 3}, S = {3, 4}

- R × S = { (1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4) }
- A relation on R, S is any subset (⊆) of R × S (e.g: { (1, 4), (3, 4)})

Database relations

Given attribute domains

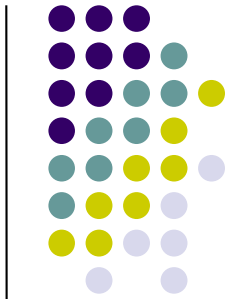
Branches = { Downtown, Brighton, … }
Accounts = { A-101, A-201, A-217, … }
Balances = R

Account ⊆ Branches × Accounts × Balances

{ (Downtown, A-101, 500),
(Brighton, A-201, 900),
(Brighton, A-217, 500) }

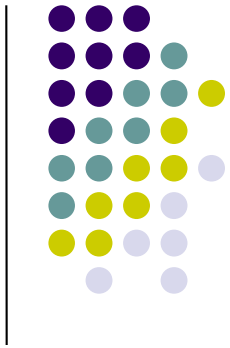
bname	acct_no	balance
Downtown	A-101	500
Brighton	A-201	900
Brighton	A-217	500

Review: Terms and Definitions



1. Tables = Relations
2. Columns = Attributes
3. Rows = Tuples
4. Relation Schema (or Schema)
    1. A list of attributes and their domains
    2. We will require the domains to be atomic
    3. E.g. account(account-number, branch-name, balance)
5. Relation Instance
    1. A particular instantiation of a relation with actual values
    2. Will change with time

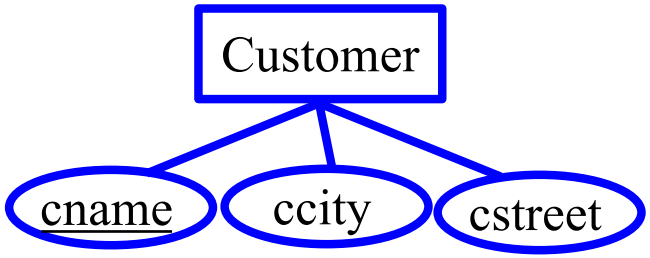
So…



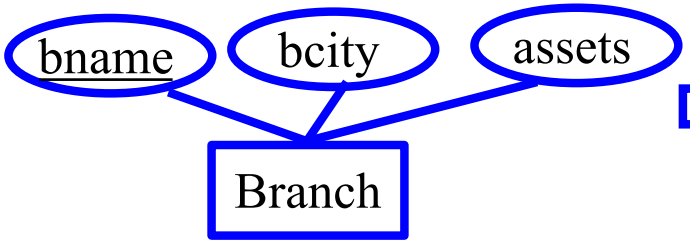
- That's the basic relational model
- That's it ?
  - What about the constraints ?
  - How do we represent one-to-one vs many-to-one relationships ?
  - Many of those constraints get embedded in the schema
    - Especially relationship cardinality constraints
  - Others are explicitly represented using other constructs

E/R Diagrams → Relations

Convert entity sets into a relational schema with the same set of attributes



Customer_Schema(cname, ccity, cstreet)

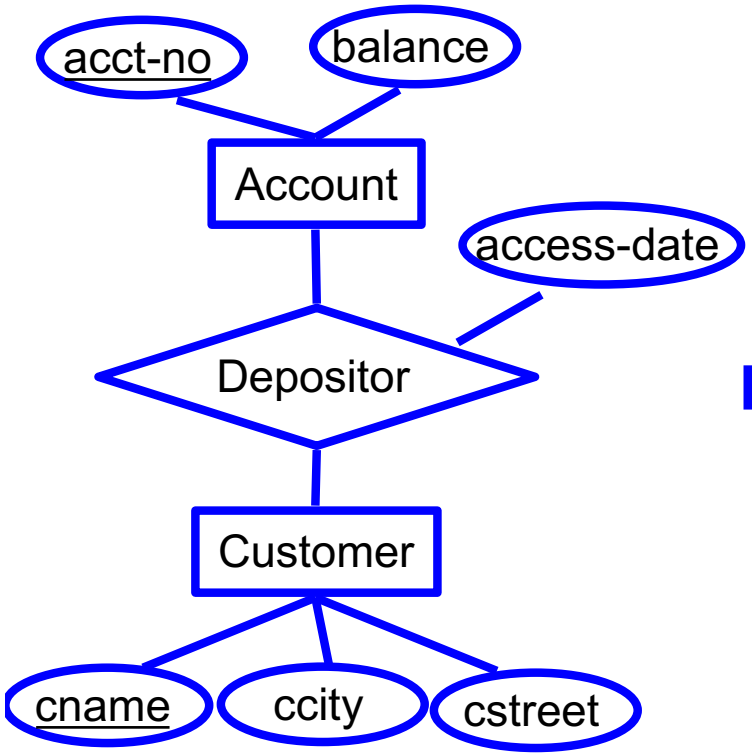


Branch_Schema(bname, bcity, assets)

E/R Diagrams → Relations

Convert relationship sets also into a relational schema

Remember: A relationship is completely described by primary keys of associate entities and its own attributes



Account_Schema(acct-no, balance)

Depositor_Schema(cname, acct-no, access-date)

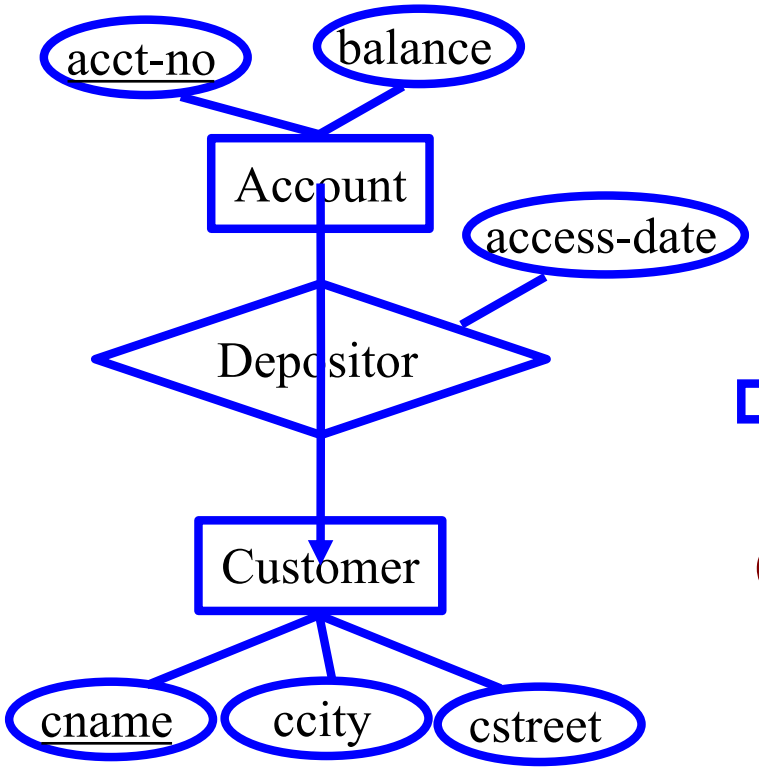
Customer_Schema(cname, ccity, cstreet)

Well… Not quite. We can do better.
It depends on the relationship cardinality

E/R Diagrams → Relations

Say One-to-Many Relationship from Customer to Account

→ Many accounts per customer

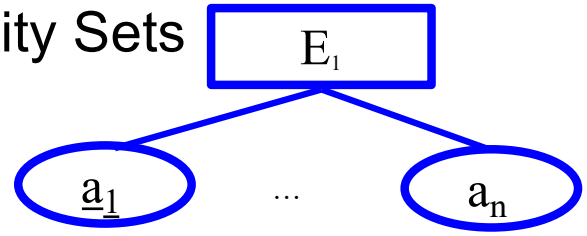


Account_Schema(acct-no, balance, cname, access-date)

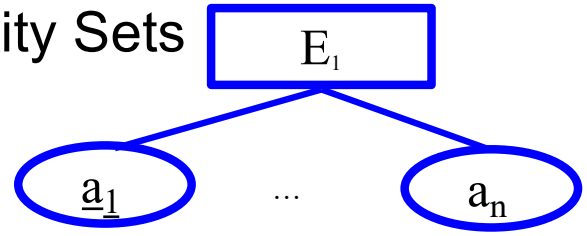
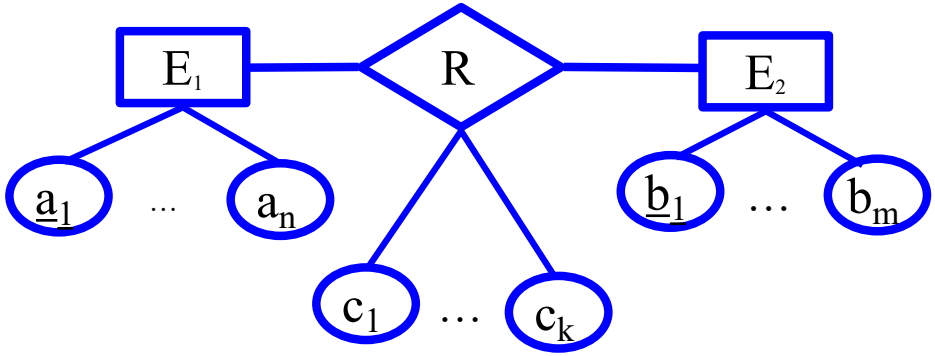
Customer_Schema(cname, ccity, cstreet)

Exactly same information, fewer tables

E/R Diagrams → Relations

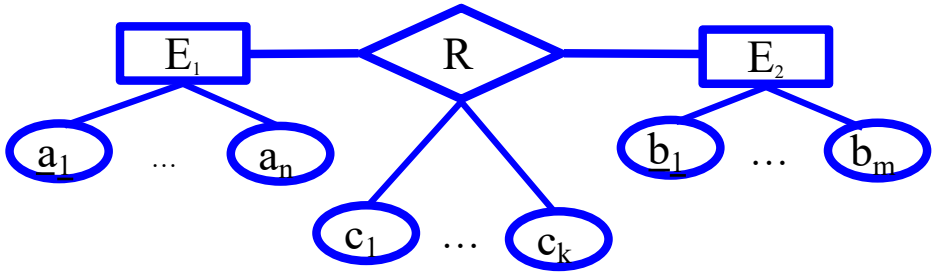
E/R	Relational Schema
Entity Sets 	$E = (\underline{a_1}, \ldots, a_n)$

E/R Diagrams → Relations

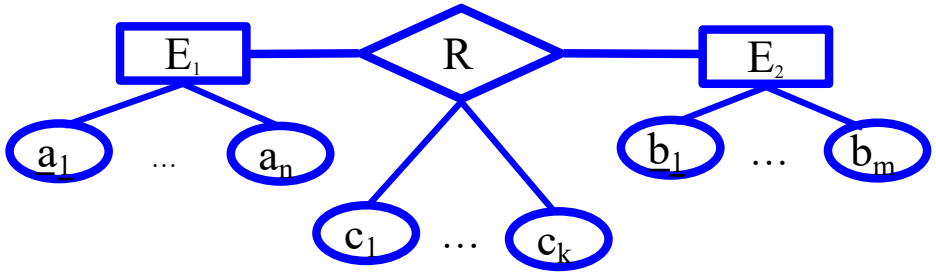
E/R	Relational Schema
Entity Sets 	$E = (\underline{a_1}, \ldots, a_n)$
Relationship Sets 	$R = (\underline{a_1}, \underline{b_1}, c_1, \ldots, c_n)$ a_1: E_1's key b_1: E_2's key $c_1, \ldots, c_k$: attributes of R

Not the whole story for Relationship Sets …

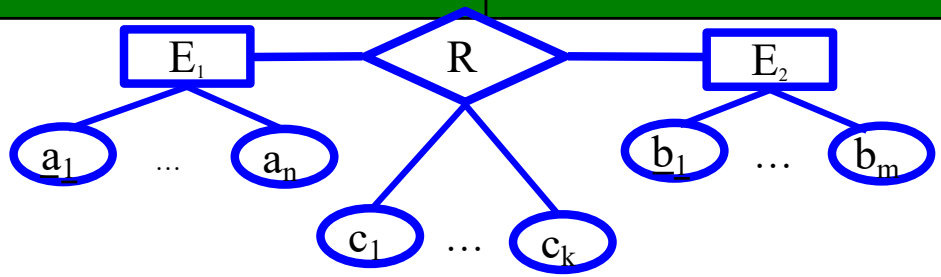
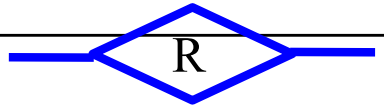
E/R Diagrams → Relations

Relationship Cardinality	Relational Schema
	
n:m 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$ $R = (\underline{a_1}, \underline{b_1}, c_1, \ldots, c_n)$

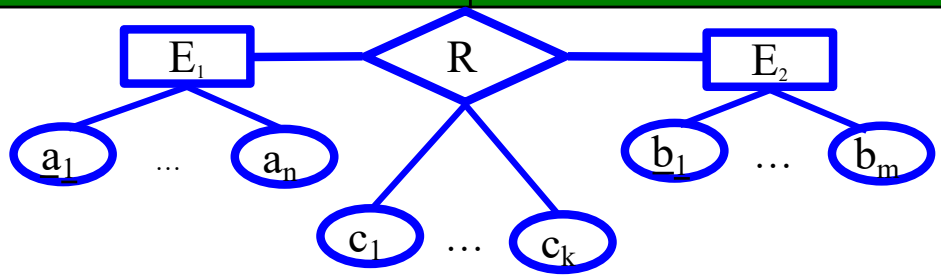
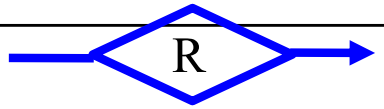
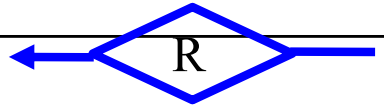
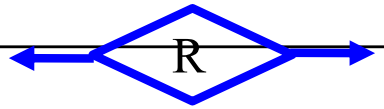
E/R Diagrams → Relations

Relationship Cardinality	Relational Schema
	
n:m 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$ $R = (\underline{a_1}, \underline{b_1}, c_1, \ldots, c_n)$
n:1 	$E_1 = (\underline{a_1}, \ldots, a_n, b_1, c_1, \ldots, c_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$

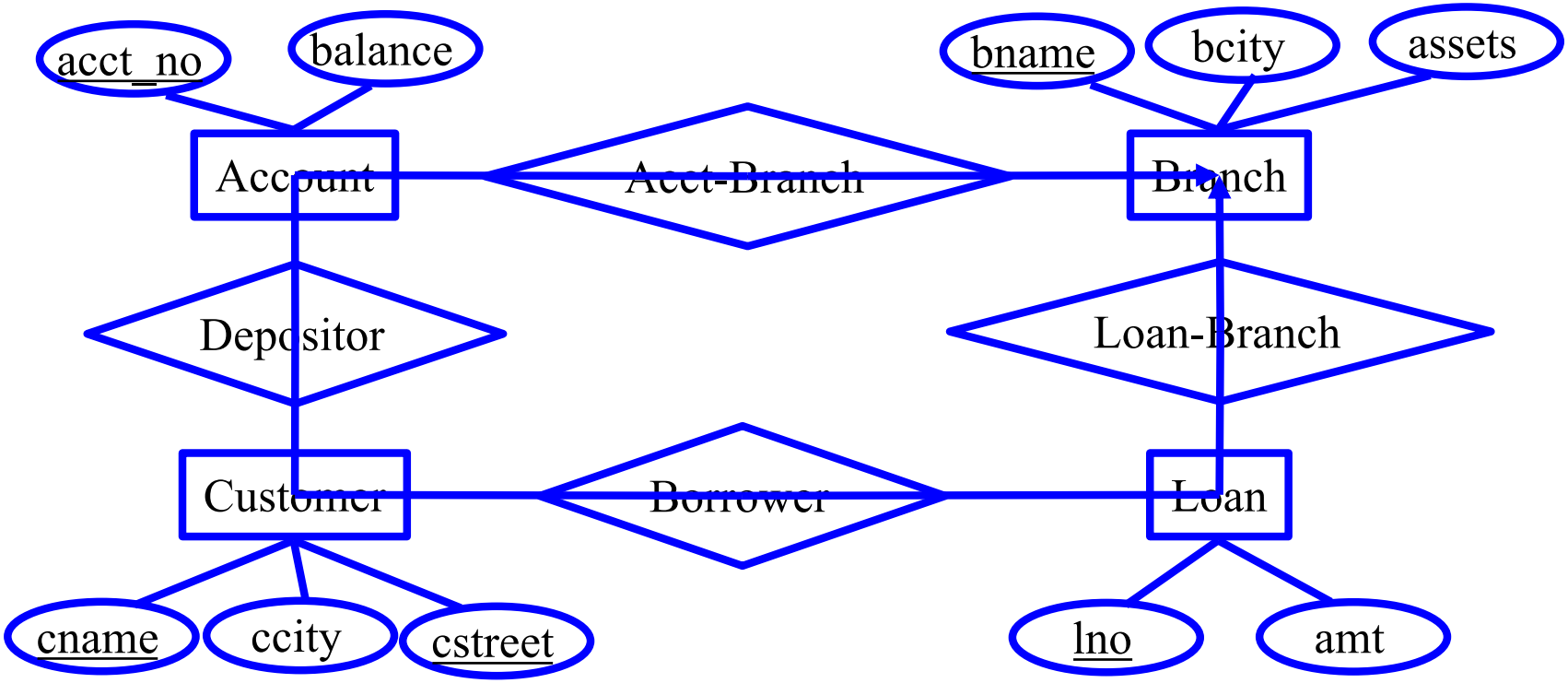
E/R Diagrams → Relations

Relationship Cardinality	Relational Schema
	
n:m  	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$ $R = (\underline{a_1}, \underline{b_1}, c_1, \ldots, c_n)$
n:1 	$E_1 = (\underline{a_1}, \ldots, a_n, b_1, c_1, \ldots, c_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$
1:n	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m, a_1, c_1, \ldots, c_n)$

E/R Diagrams → Relations

Relationship Cardinality	Relational Schema
	
n:m 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$ $R = (\underline{a_1}, \underline{b_1}, c_1, \ldots, c_n)$
n:1 	$E_1 = (\underline{a_1}, \ldots, a_n, b_1, c_1, \ldots, c_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$
1:n 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m, a_1, c_1, \ldots, c_n)$
1:1 	Treat as n:1 or 1:n

Translating E/R Diagrams to Relations



Q. How many tables does this get translated into?

A. 6 (account, branch, customer, loan, depositor, borrower)

Bank Database

Account

bname	acct_no	balance
Downtown	A-101	500
Mianus	A-215	700
Perry	A-102	400
R.H.	A-305	350
Brighton	A-201	900
Redwood	A-222	700
Brighton	A-217	750

Depositor

cname	acct_no
Johnson	A-101
Smith	A-215
Hayes	A-102
Turner	A-305
Johnson	A-201
Jones	A-217
Lindsay	A-222

Customer

cname	cstreet	ccity
Jones	Main	Harrison
Smith	North	Rye
Hayes	Main	Harrison
Curry	North	Rye
Lindsay	Park	Pittsfield
Turner	Putnam	Stanford
Williams	Nassau	Princeton
Adams	Spring	Pittsfield
Johnson	Alma	Palo Alto
Glenn	Sand Hill	Woodside
Brooks	Senator	Brooklyn
Green	Walnut	Stanford

Branch

bname	bcity	assets
Downtown	Brooklyn	9M
Redwood	Palo Alto	2.1M
Perry	Horseneck	1.7M
Mianus	Horseneck	0.4M
R.H.	Horseneck	8M
Pownel	Bennington	0.3M
N. Town	Rye	3.7M
Brighton	Brooklyn	7.1M

Borrower

cname	lno
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17
Adams	L-16

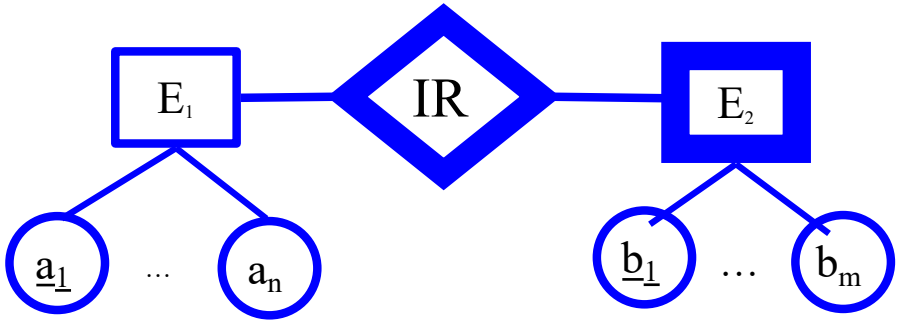
Loan

bname	lno	amt
Downtown	L-17	1000
Redwood	L-23	2000
Perry	L-15	1500
Downtown	L-14	1500
Mianus	L-93	500
R.H.	L-11	900
Perry	L-16	1300

E/R Diagrams & Relations

E/R	Relational Schema

Weak Entity Sets



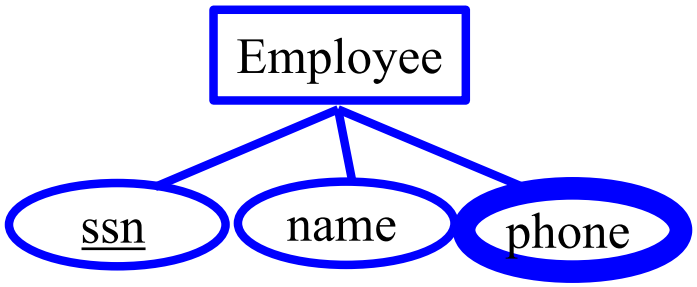
$E_1 = (\underline{a_1}, \ldots, a_n)$

$E_2 = (\underline{a_1, b_1}, \ldots, b_m)$

E/R Diagrams & Relations

E/R	Relational Schema

Multivalued Attributes



Emp = (ssn, name)
Emp-Phones = (ssn, phone)

ssn	name
001	Smith
…	…

Emp

ssn	phone
001	4-1234
001	4-5678
…	…

Emp-Phones

How to create Tables

- The relationship model represents the database as a collection of relations.
- A relation is made up of 2 parts:

1. Relation instance: a table, with rows and columns.
- (columns = fields = attribute)(rows = tuples)
- 2. Relation scheme: specifies the name of relation, plus name and domain of each field.

The relation schema is: Customers (customer-name, customer-street)

customer-name	customer-street
US-Robotics	Main
Westing	North
US-Robotics	North

Basic Structure

- We also represent the schema in the following way:

Customer	customer-name	customer-street

- The notions of superkey, candidate keys and primary key are
- applicable to a relation.
- The primary key is underlined in the above.

We can think of a relation as a set of rows or tuples.

A key can uniquely identify the tuples.

Schema Definition in SQL

Customer	customer-name	customer-street	customer-city

- Create table customer
- (customer-name char(20) not null,
- customer-street char(30),
- customer-city char(30),
- primary key (customer-name))

Schema Definition in SQL

- To remove a relation from an SQL database, we use the
- drop table command: drop table r.

- We use the alter table command to add or delete attributes to an existing relation.

- All tuples in the relation are assigned null for a new attribute.

    - alter table customer add phone char(10)
        - alter table customer drop phone

Integrity Contraints (ICs)

- IC: condition that must be true for any instance of the database.
- E.g. The not null constraint means the value of the attribute cannot be null.
- ICs are specified when the schema is defined.
- ICs are checked when relations are modified.

Domain Constraints

- create domain hourly-wage numeric(5,2)
- constraint wage-value-test check (value > 4.00)

Constraint name (optional): if the constraint is violated,
the constraint name is returned and can be used to identify the error

- The domain hourly-wage is a decimal number with 5 digits,
- 2 of which are placed after the decimal point.
- The domain has a constraint that ensures that the hourly wage is greater than 4.00.

- In SQL, we can declare
  - a key by the UNIQUE command
  - a primary key by the PRIMARY KEY constraint.

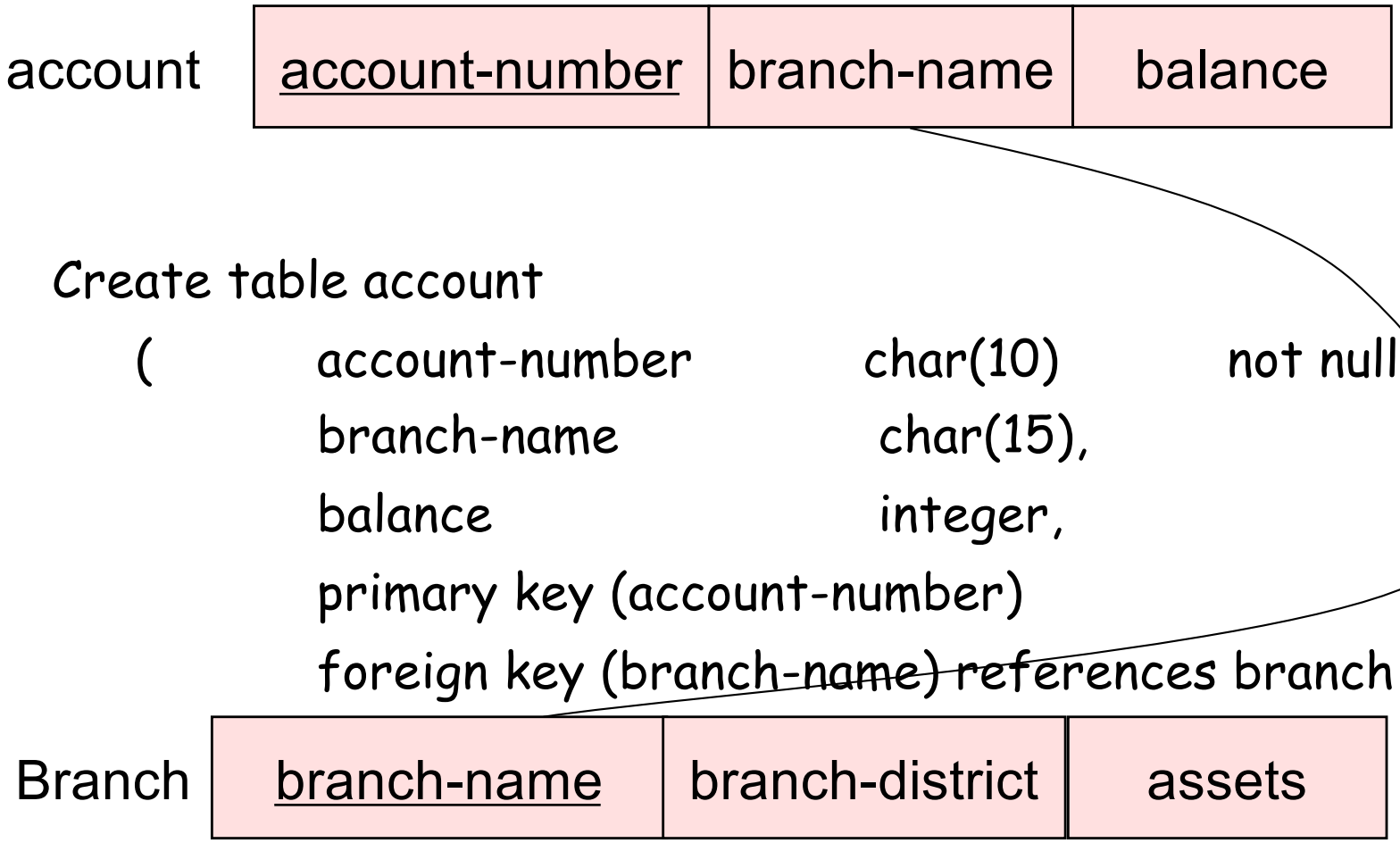
```
CREATE TABLE Students
        (sid     CHAR(20),
         name    CHAR(30),
         login   CHAR(20),
         age     INTEGER,
         gpa     REAL,
         UNIQUE (name, age),
         CONSTRAINT StudentsKey PRIMARY KEY ( sid))
```

Primary key

key

Constraint name (optional)

Foreign Keys, Referential Integrity



- Create table account
- (account-number char(10) not null,
- branch-name char(15),
- balance integer,
- primary key (account-number)
- foreign key (branch-name) references branch)

Foreign Keys, Referential Integrity

- A foreign key is a set of fields in one relation r that is used
- to refer to a tuple in another relation s. (It must
- correspond to the primary key of the second relation.)

- The foreign key condition specifies a referential integrity
- constraint.

Foreign Keys, Referential Integrity

Depositor	customer-name	account-number

- Create table depositor
- (customer-name char(20) not null,
- account-number char(10) not null,
- primary key (customer-name, account-number),
- foreign key (customer-name) references customer
- foreign key (account-number) references account)

cannot add (KLN, 111, 3) to Account.

- Example: cannot add (Chris, 222) to Depositor.

  cannot add (Mary, 999) to Depositor.

Customer

customer-name	customer-street
Tim	Main
Smith	North
Mary	North

Branch

branch-name	branch-district	assets
CUHK	Shatin	1000
CMTR	Central	2000
SKCR	Shatin	4000

Account

branch-name	account-number	balance
CUHK	222	2
CUHK	777	5
CMTR	333	1
SKCR	444	5
SKCR	888	5

Depositor

customer-name	account-number
Tim	222
Mary	888
Tim	444
Smith	333
Tim	333

- Example:

Cannot simply delete Customer Tim,

Cannot simply delete branch CUHK

Customer

customer-name	customer-street
Tim	Main
Smith	North
Mary	North

Branch

branch-name	branch-district	assets
CUHK	Shatin	1000
CMTR	Central	2000
SKCR	Shatin	4000

Account

branch-name	account-number	balance
CUHK	222	2
CUHK	777	5
CMTR	333	1
SKCR	444	5
SKCR	888	5

Depositor

customer-name	account-number
Tim	222
Mary	888
Tim	444
Smith	777
Tim	333
Smith	888

Foreign Keys, Referential Integrity

- Cascading delete:
- Delete Customer Tim, cascaded delete on depositor
- Delete branch CUHK, cascaded delete on account

```
create table account
    (branch-name        char(15),
     account-number             char(10)    not null,
     balance            integer,
     primary key (account-number),
     foreign key (branch-name)      references branch
                                    on delete cascade
                                    on update cascade )
```