# Database Design Steps

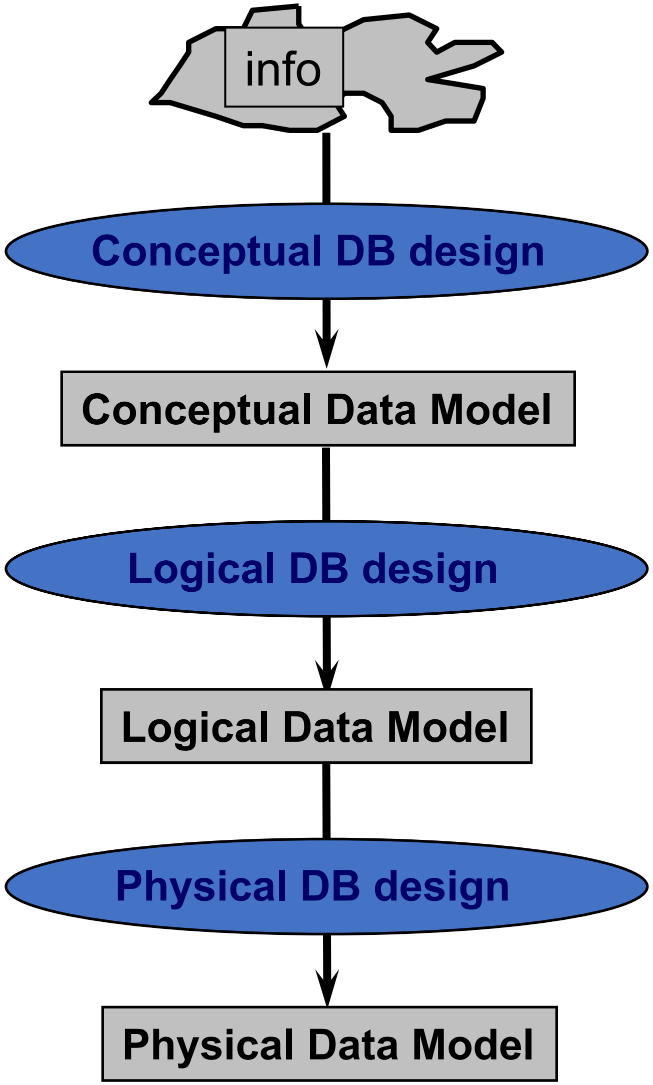
**Entity-relationship Model**

Typically used for conceptual database design

Three Levels of Modeling

**Relational Model**

Typically used for logical database design



info

Conceptual DB design

Conceptual Data Model

Logical DB design

Logical Data Model

Physical DB design

Physical Data Model

# Relational Data Model

## Introduced by Ted Codd (late 60’s – early 70’s)

- *Before = “Network Data Model” (Cobol as DDL, DML)*
- *Very contentious: Database Wars (Charlie Bachman vs.*
  *Mike Stonebraker)*

## Relational data model contributes:

1. *Separation of logical, physical data models (data independence)*
2. *Declarative query languages*
3. *Formal semantics*
4. *Query optimization (key to commercial success)*

## 1st prototypes:

- *Ingres → CA*
- *Postgres → Illustra → Informix → IBM*
- *System R → Oracle, DB2*

# Key Abstraction: Relation

Account =

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

Terms:

- Tables (aka: Relations)

*Why called Relations?*

# Why Called Relations?

## Mathematical relations

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

- $R \times S = \{ (1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4) \}$
- *A* ***relation*** *on R, S is any subset (*$\subseteq$*) of* $R \times S$

  *(e.g:* $\{ (1, 4), (3, 4)\}$*)*

## Database relations

*Given attribute domains*

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

*Account* $\subseteq$ *Branches* $\times$ *Accounts* $\times$ *Balances*

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

# Relations

Account =

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

*Considered equivalent to…*

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

*Relational database semantics defined in terms of mathematical relations*

# So…

- That’s the basic relational model
- Some more questions/issues
  - What about semantic information ?
    - Relationships between entities ?
  - What about the constraints ?
  - How do we represent one-to-one vs many-to-one relationships ?
  - Those constraints are all embedded in the schema

# Keys and Relations

- Keys: Sets of attributes that allow us to identify entities
- Very loosely speaking, tuples are viewed entities

# Keys

- Superkey
  - set of attributes of table for which every row has distinct set of values
- Candidate key
  - Minimal such set of attributes
- Primary key
  - DB Chosen Candidate key
  - Plays a very important role
    - E.g. relations typically sorted by this

# Customer Table

| Customer | | | | | | |
|---|---|---|---|---|---|---|
| Cust-id | Cust-name | DOB | Phone | Age | Cust-street | Cust-City |

**Possible Keys:**

{cust-id}

{cust-name, cust-city, cust-street}

{cust-id, age}

{cust-name, phone}

cust-name ?? Probably not.

**Domain knowledge dependent !!**

# Example of Keys

- *Superkey*
  - any attribute set that can distinguish entities
- *Candidate key*
  - a minimal superkey
    - Can’t remove any attribute and preserve key-ness
      - {cust-id, age} not a candidate key since we could have {cust-id} by removing “age” of {cust-id, age}
      - {cust-name, cust-city, cust-street} is
        - assuming cust-name is not unique
- *Primary key*
  - Candidate key chosen as *the* key by DBA

# Keys

- Also act as integrity constraints
  - i.e., guard against illegal/invalid instance of given schema

e.g., Branch = ($\underline{\text{bname}}$, bcity, assets)

| bname | bcity | assets |
|---|---|---|
| Brighton | Brooklyn | 5M |
| Brighton | Boston | 3M |

*Invalid*

# More on Keys

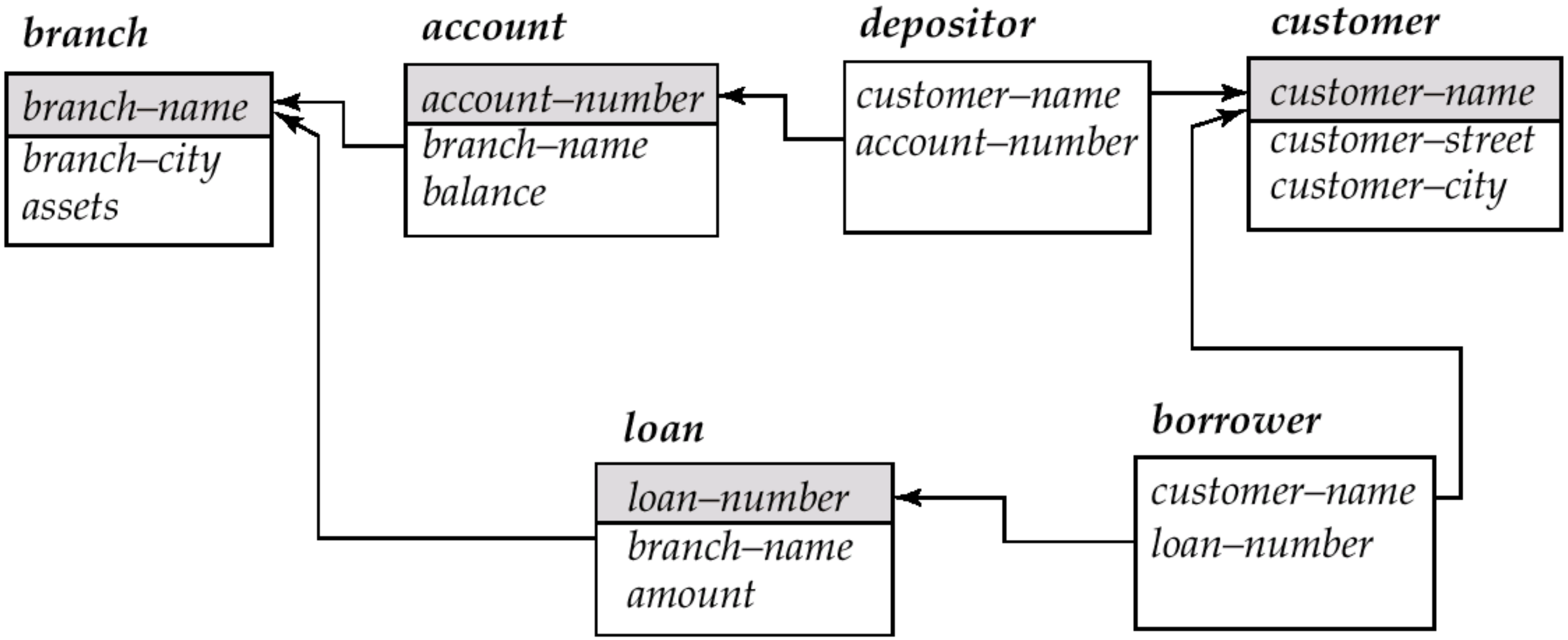
## Determining Primary Keys

Find candidate keys (minimal sets of attributes that can uniquely identify a tuple) and pick up one as primary key

## Foreign Keys

If a relation schema includes the primary key of another relation schema, that attribute is called the *foreign key*

# Schema Diagram for the Banking Enterprise



***branch***

*branch–name*
*branch–city*
*assets*

***account***

*account–number*
*branch–name*
*balance*

***depositor***

*customer–name*
*account–number*

***customer***

*customer–name*
*customer–street*
*customer–city*

***loan***

*loan–number*
*branch–name*
*amount*

***borrower***

*customer–name*
*loan–number*