雨课堂
Rain Classroom

本次课程是

线上+线下
融合式教学

请**现场**的同学们：

1. 打开雨课堂，点击页面右下角喇叭按钮调至静音状态

请**远程上课**的同学们：

1. 打开雨课堂，点击页面右下角喇叭按钮调至静音状态
2. 打开“瞩目”（会议室：182 943 865；密码：见学堂公告），进入会议室，并关闭麦克风

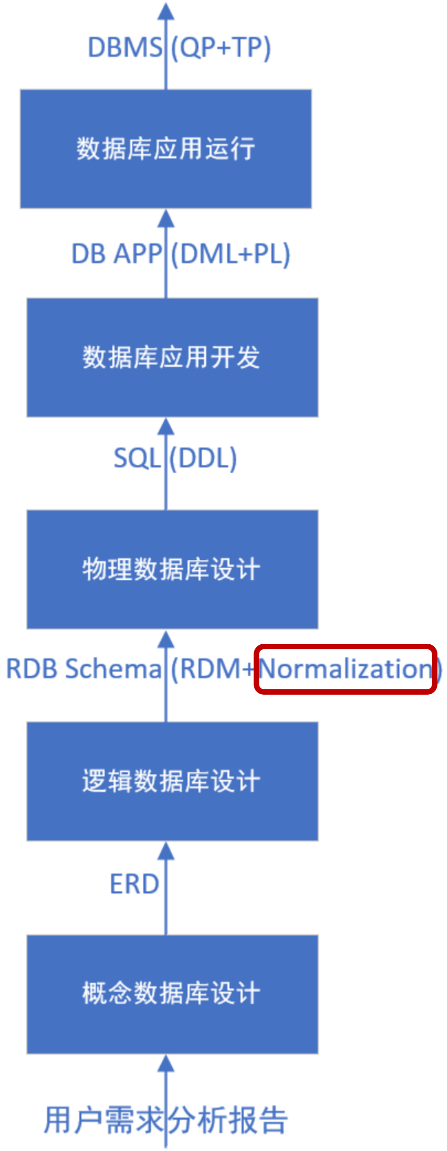
Database Concepts (IV)

Database Analysis and Design

Chaokun Wang

School of Software, Tsinghua University
chaokun@tsinghua.edu.cn

April 11, 2022



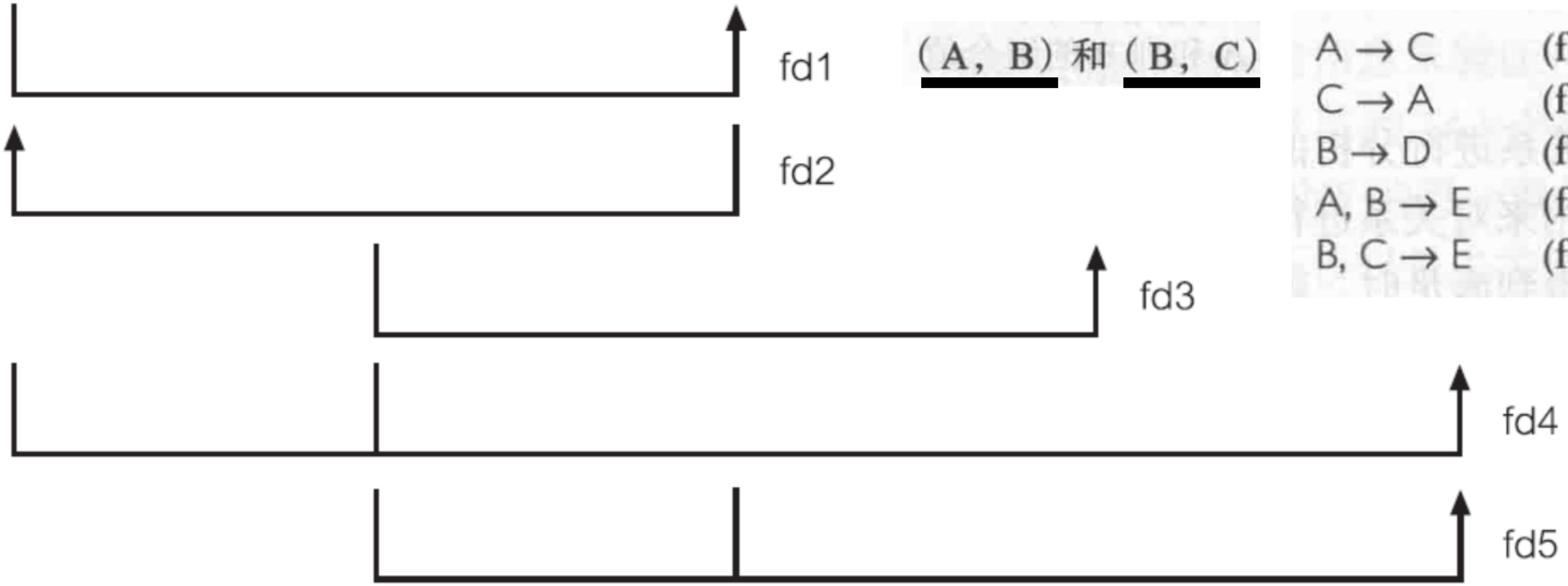
Outline

- DB Development Lifecycle
- Entity-relationship Modeling*
- ✈ Database Normalization*
- Database Design

Sample Relation with Functional Dependencies (FD)

Sample Relation

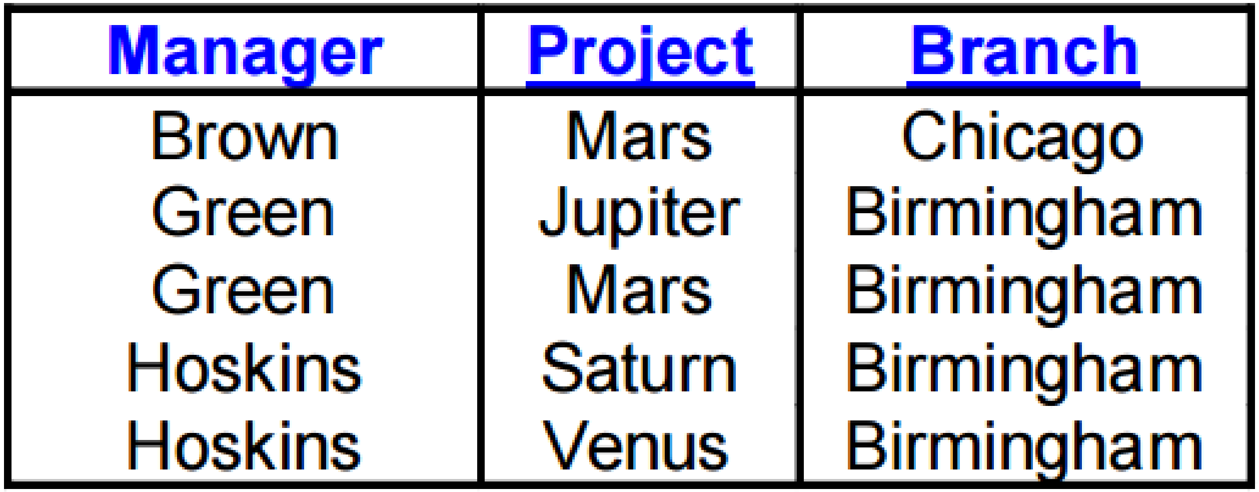
A	B	C	D	E
a	b	z	w	q
e	b	r	w	p
a	d	z	w	t
e	d	r	w	q
a	f	z	s	t
e	f	r	s	t



(A，B) 和 (B，C)

A → C	(fdl)
C → A	(fd2)
B → D	(fd3)
A, B → E	(fd4)
B, C → E	(fd5)

Relation with FD



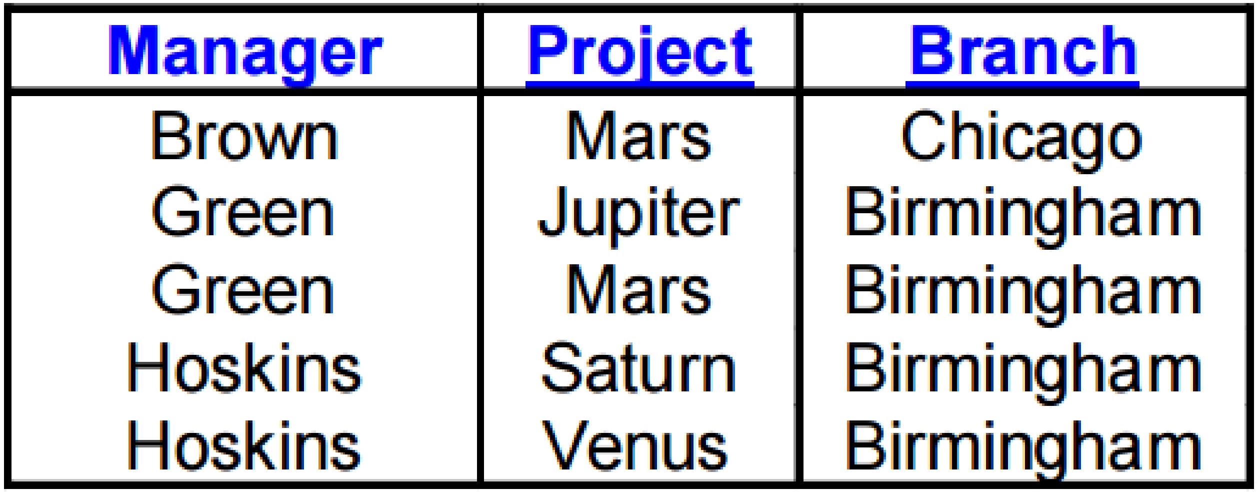
Manager	Project	Branch
Brown	Mars	Chicago
Green	Jupiter	Birmingham
Green	Mars	Birmingham
Hoskins	Saturn	Birmingham
Hoskins	Venus	Birmingham

- fd1: Manager -> Branch
- fd2: Project, Branch -> Manager

- primary key: (Project, Branch)

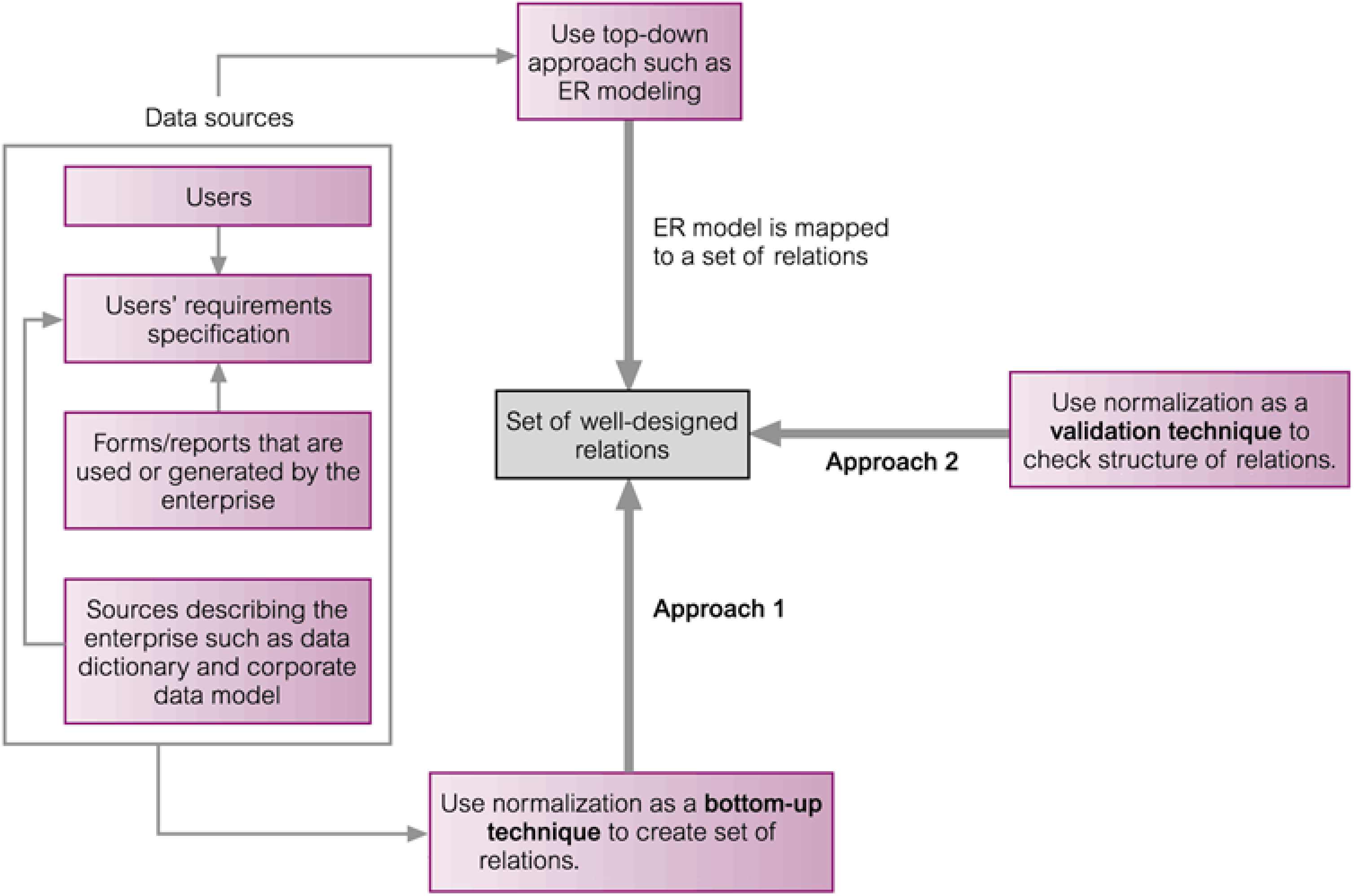
Boyce–Codd Normal Form (BCNF)

- A relation is in BCNF if and only if every determinant is a candidate key
  - – For every non-trivial functional dependency $Y \rightarrow Z$, Y is a candidate key.



Manager	Project	Branch
Brown	Mars	Chicago
Green	Jupiter	Birmingham
Green	Mars	Birmingham
Hoskins	Saturn	Birmingham
Hoskins	Venus	Birmingham

How Normalization Supports Database Design


Use top-down approach such as ER modeling
Data sources
Users
Users' requirements specification
Forms/reports that are used or generated by the enterprise
Sources describing the enterprise such as data dictionary and corporate data model
ER model is mapped to a set of relations
Set of well-designed relations
Use normalization as a **validation technique** to check structure of relations.
Approach 2
Approach 1
Use normalization as a **bottom-up technique** to create set of relations.

Tutorial: Normalization Data Table

User ID	Customer Name	Order Product	Total Price	Card Type	Card No.	Billing Address	Shipment Address	Purchase Time
1	Sheldon Cooper	Agents of Atlas #1, $3.99 Alien Legion #19, $2.99	6.98	Visa	0622 1234 5678 4321	350 Fifth Avenue, New York, NY 10118-3299	350 Fifth Avenue, New York, NY 10118-3299	2019-11-11 09:36:45
1	Sheldon Cooper	Agents of Atlas #2, $3.99 Alien Legion #20, $2.99	6.98	Master Card	0543 1234 4321 9876	350 Fifth Avenue, New York, NY 10118-3299	1145 17th Street NW, Washington, D.C. 20090-8199	2019-11-12 19:45:32
2	Howard Wolowitz	Avengers A.I. #1, $0.99 Alien Legion #20, $1.99	2.98	Master Card	4321 7777 5332 1986	1145 17th Street NW, Washington, D.C. 20090-8199	1145 17th Street NW, Washington, D.C. 20090-8199	2019-11-14 14:32:12
3	Leonard Hofstadter	Dark Angel #16, $5.99	5.99	Visa	1735 8973 4578 2975	1200 West Harrison Street, Chicago, Illinois 60607-7161	1200 West Harrison Street, Chicago, Illinois 60607-7161	2019-11-15 15:04:03

Tutorial: Normalization Data Table

User ID
Last Name
First Name
Series
Issue
Price
Total Price
Card Type
Card No
Street (Billing)
City (Billing)
State (Billing)
Postcode (Billing)
Street (Shipping)
City (Shipping)
State (Shipping)
Postcode (Shipping)
Purchase Time

Order ID
User ID
Last Name
First Name
Prod ID
Series
Issue
Price
Total Price
Card Type
Card No
Street (Billing)
……
Street (Shipping)
……
Purchase Time

Order ID
User ID
Last Name
First Name
Total Price
Card Type
Card No
Street (Billing)
……
Street (Shipping)
……
Purchase Time

Prod ID
Series
Issue
Price

Order_Prod ID
Order ID
Prod ID

Tutorial: Normalization Data Table

Order ID
User ID
Last Name
First Name
Total Price
Card Type
Card No
Street (Billing)
City (Billing)
State (Billing)
Postcode (Billing)
Street (Shipping)
City (Shipping)
State (Shipping)
Postcode (Shipping)
Purchase Time

Order ID
User ID
Total Price
Card Type
Card No
Street (Billing)
City (Billing)
State (Billing)
Postcode (Billing)
Street (Shipping)
City (Shipping)
State (Shipping)
Postcode (Shipping)
Purchase Time

User ID
Last Name
First Name

Address ID
User ID
Street
City
State
Postcode

Pay Method ID
User ID
Card Type
Card No

Prod ID
Series
Issue
Price

Order_Prod_ID
Order ID
Prod ID

Order ID
User ID
Total Price
Pay Method ID
Billing Addr ID
Shipping Addr ID
Purchase Time

Tutorial: Normalization Data Table

City ID
City
State ID

State ID
State

User ID
Last Name
First Name

Address ID
User ID
Street
City ID
Postcode

Pay Method ID
User ID
Card Type
Card No

Prod ID
Series
Issue
Price

Order_Prod_ID
Order ID
Prod ID

Order ID
User ID
Total Price
Pay Method ID
Billing Addr ID
Shipping Addr ID
Purchase Time

Codd's Relational Database Rules (1 of 2)

Table 13.8	Dr. Codd’s 12 Relational Database Rules	
Rule	Rule Name	Description
1	Information	All information in a relational database must be logically represented as column values in rows within tables.
2	Guaranteed access	Every value in a table is guaranteed to be accessible through a combination of table name, primary key value, and column name.
3	Systematic treatment of nulls	Nulls must be represented and treated in a systematic way, independent of data type.
4	Dynamic online catalog based on the relational model	The metadata must be stored and managed as ordinary data—that is, in tables within the database; such data must be available to authorized users using the standard database relational language.
5	Comprehensive data sublanguage	The relational database may support many languages; however, it must support one well-defined, declarative language as well as data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorization, and transaction management (begin, commit, and rollback).
6	View updating	Any view that is theoretically updatable must be updatable through the system.
7	High-level insert, update, and delete	The database must support set-level inserts, updates, and deletes.

Codd's Relational Database Rules (2 of 2)

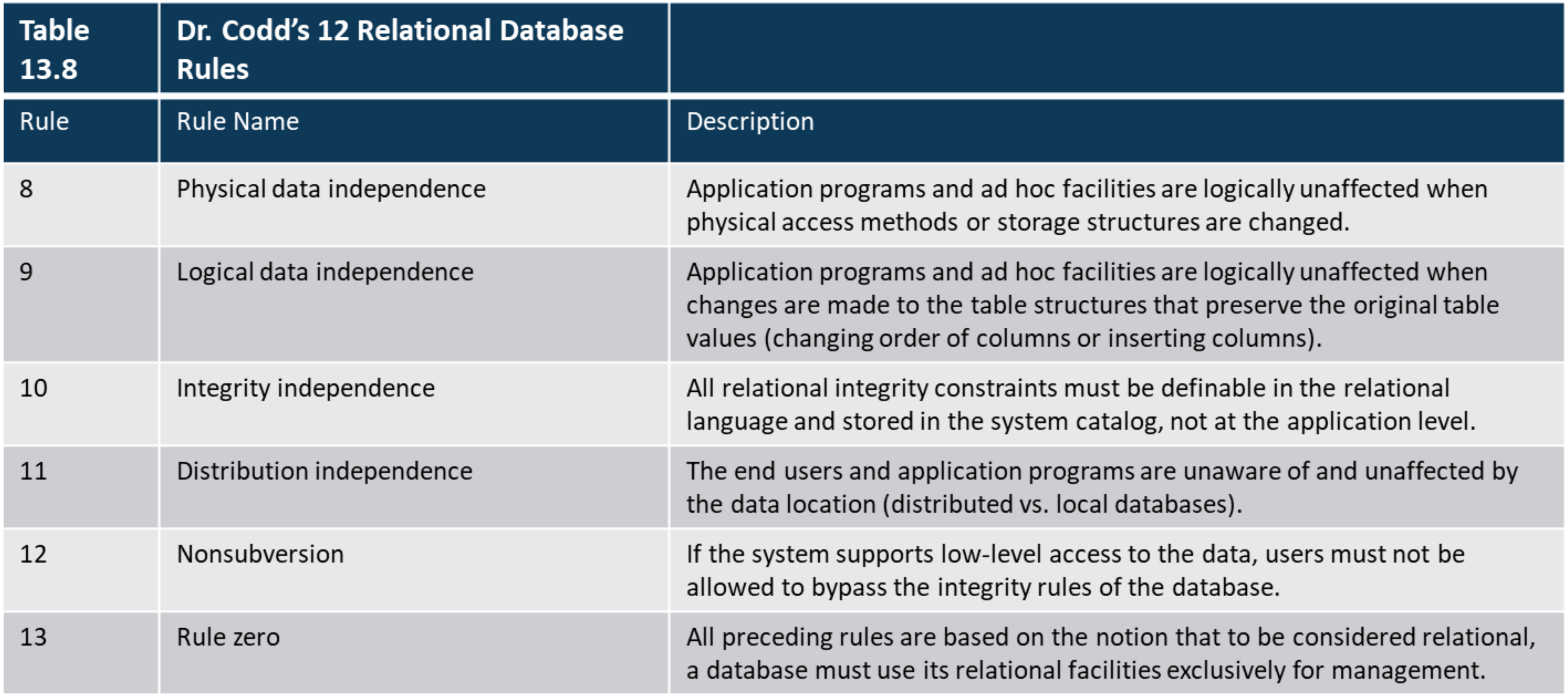
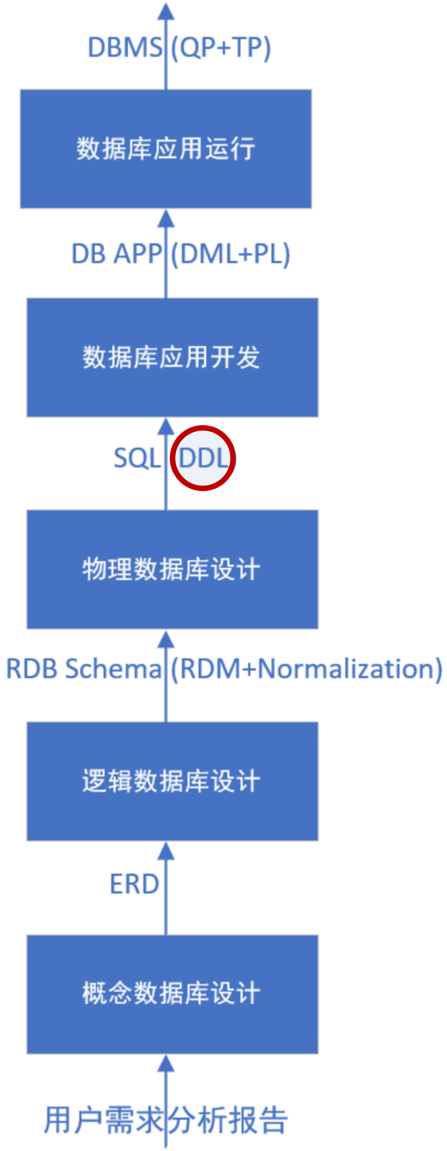


Table 13.8	Dr. Codd’s 12 Relational Database Rules	
Rule	Rule Name	Description
8	Physical data independence	Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structures are changed.
9	Logical data independence	Application programs and ad hoc facilities are logically unaffected when changes are made to the table structures that preserve the original table values (changing order of columns or inserting columns).
10	Integrity independence	All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.
11	Distribution independence	The end users and application programs are unaware of and unaffected by the data location (distributed vs. local databases).
12	Nonsubversion	If the system supports low-level access to the data, users must not be allowed to bypass the integrity rules of the database.
13	Rule zero	All preceding rules are based on the notion that to be considered relational, a database must use its relational facilities exclusively for management.



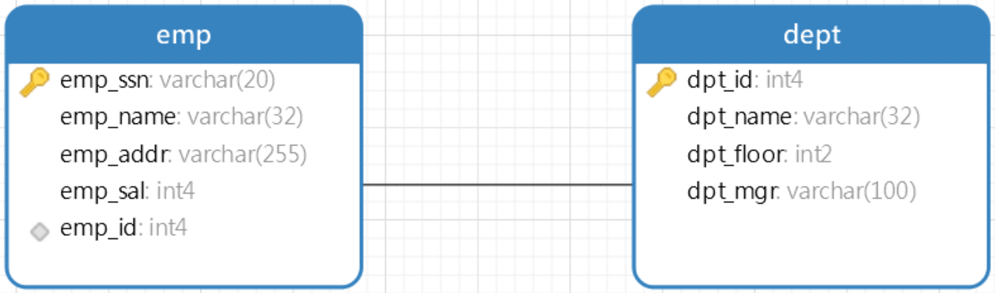
Outline

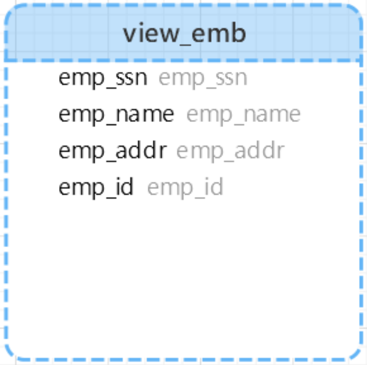
- DB Development Lifecycle
- Entity-relationship Modeling*
- Database Normalization*
- ✈ Database Design

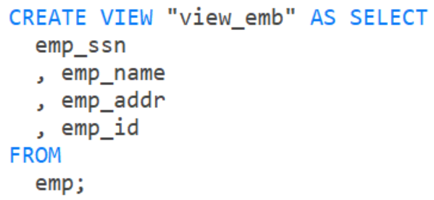
Physical Database Design Steps

- Define data storage organization
  - Design base relation
  - Choose file organizations
  - Choose indexes
  - Design user views
- Define integrity and security measures
  - Design general constraint
  - Define user and security groups and roles
  - Design security mechanisms
- Determine performance measurements
  - Analyze transactions
  - Estimate disk space requirements

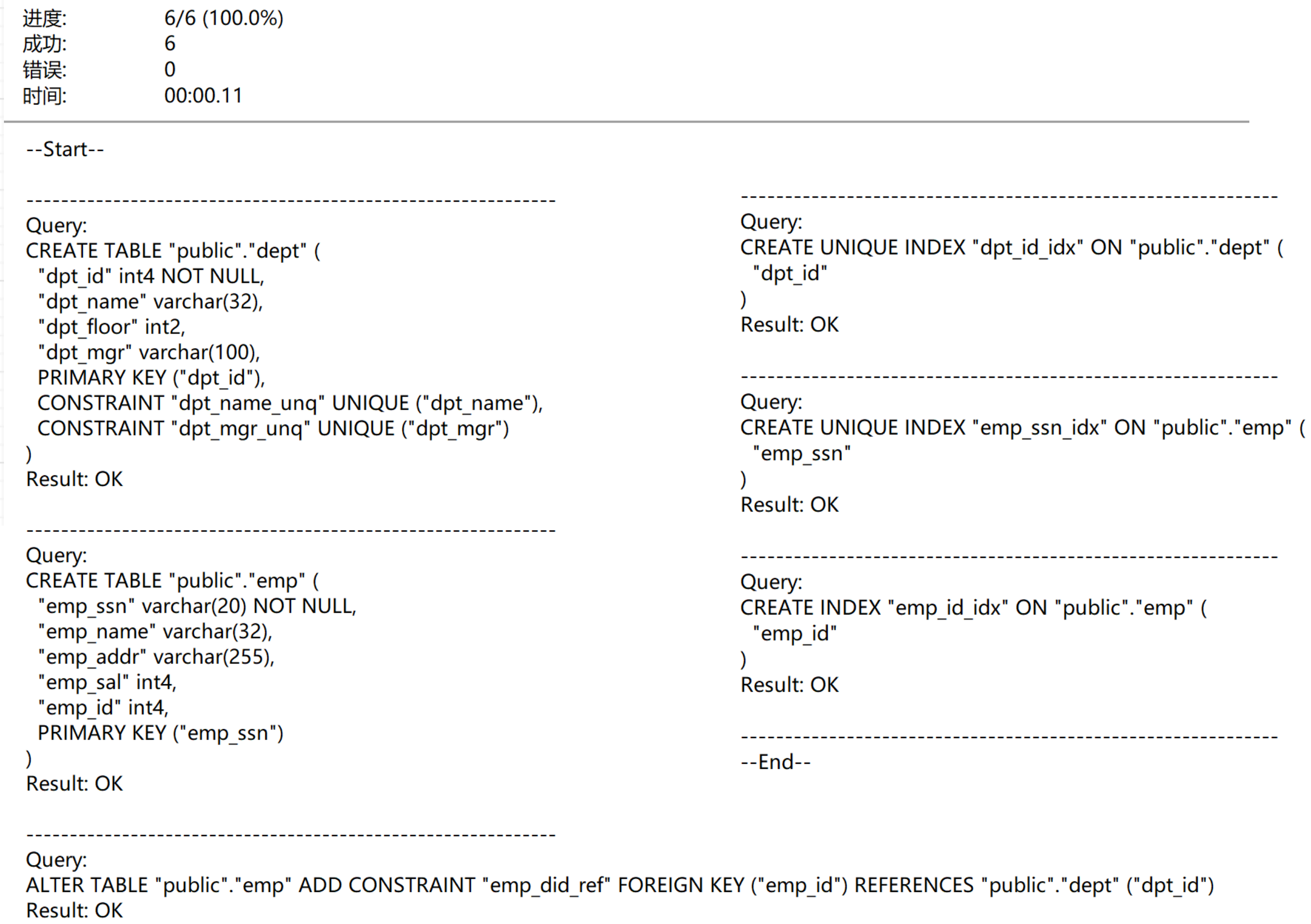
Automatic Design







```
CREATE VIEW "view_emb" AS SELECT
  emp_ssn
  , emp_name
  , emp_addr
  , emp_id
FROM
  emp;
```

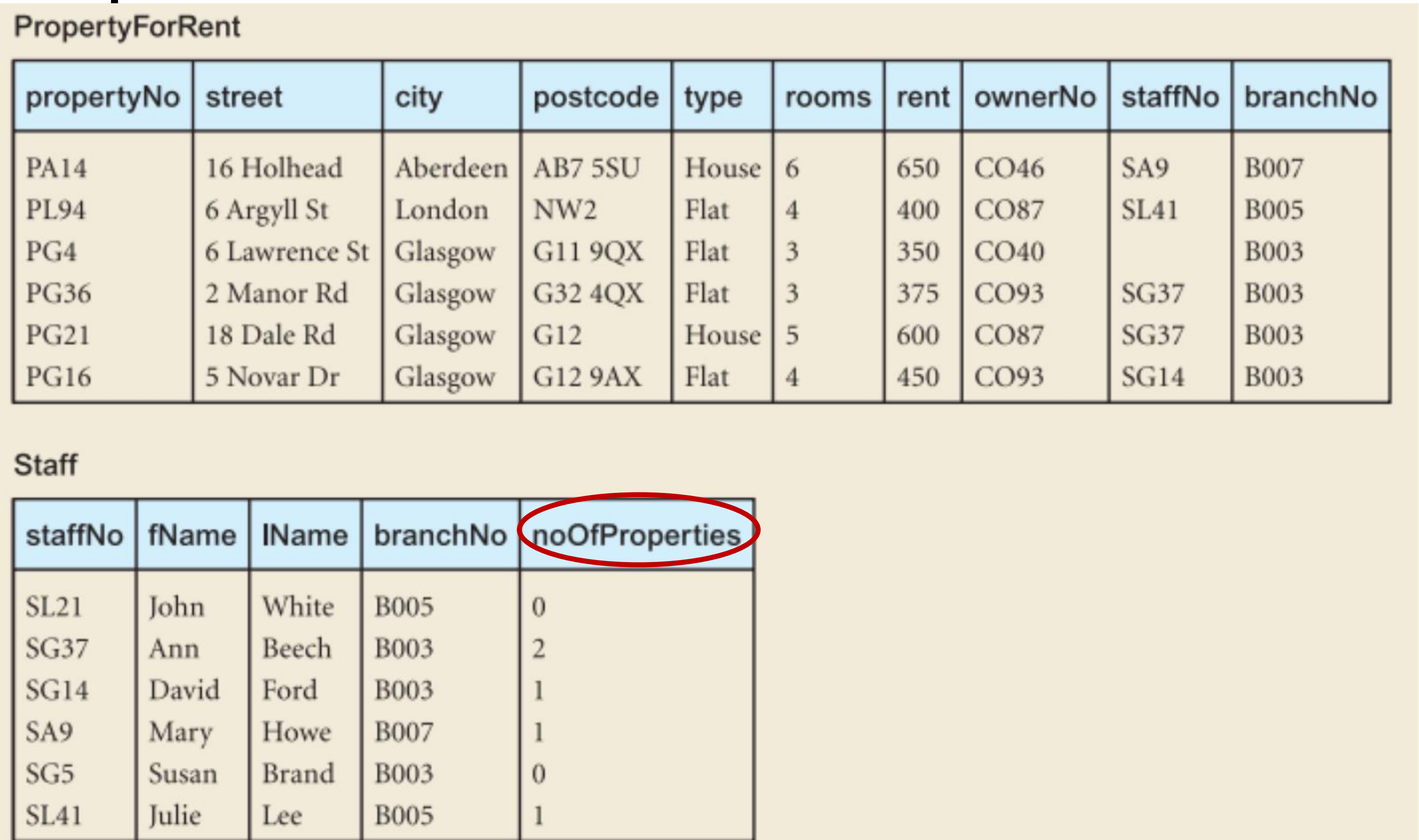


```
进度:       6/6 (100.0%)
成功:       6
错误:       0
时间:       00:00.11
```

```
--Start--

----------------------------------------------------------------
Query:
CREATE TABLE "public"."dept" (
  "dpt_id" int4 NOT NULL,
  "dpt_name" varchar(32),
  "dpt_floor" int2,
  "dpt_mgr" varchar(100),
  PRIMARY KEY ("dpt_id"),
  CONSTRAINT "dpt_name_unq" UNIQUE ("dpt_name"),
  CONSTRAINT "dpt_mgr_unq" UNIQUE ("dpt_mgr")
)
Result: OK

----------------------------------------------------------------
Query:
CREATE TABLE "public"."emp" (
  "emp_ssn" varchar(20) NOT NULL,
  "emp_name" varchar(32),
  "emp_addr" varchar(255),
  "emp_sal" int4,
  "emp_id" int4,
  PRIMARY KEY ("emp_ssn")
)
Result: OK

----------------------------------------------------------------
Query:
ALTER TABLE "public"."emp" ADD CONSTRAINT "emp_did_ref" FOREIGN KEY ("emp_id") REFERENCES "public"."dept" ("dpt_id")
Result: OK

----------------------------------------------------------------
Query:
CREATE UNIQUE INDEX "dpt_id_idx" ON "public"."dept" (
  "dpt_id"
)
Result: OK

----------------------------------------------------------------
Query:
CREATE UNIQUE INDEX "emp_ssn_idx" ON "public"."emp" (
  "emp_ssn"
)
Result: OK

----------------------------------------------------------------
Query:
CREATE INDEX "emp_id_idx" ON "public"."emp" (
  "emp_id"
)
Result: OK

----------------------------------------------------------------
--End--
```

Design representation of derived data

- The PropertyForRent relation and a simplified Staff relation with the derived attribute noOfProperties.



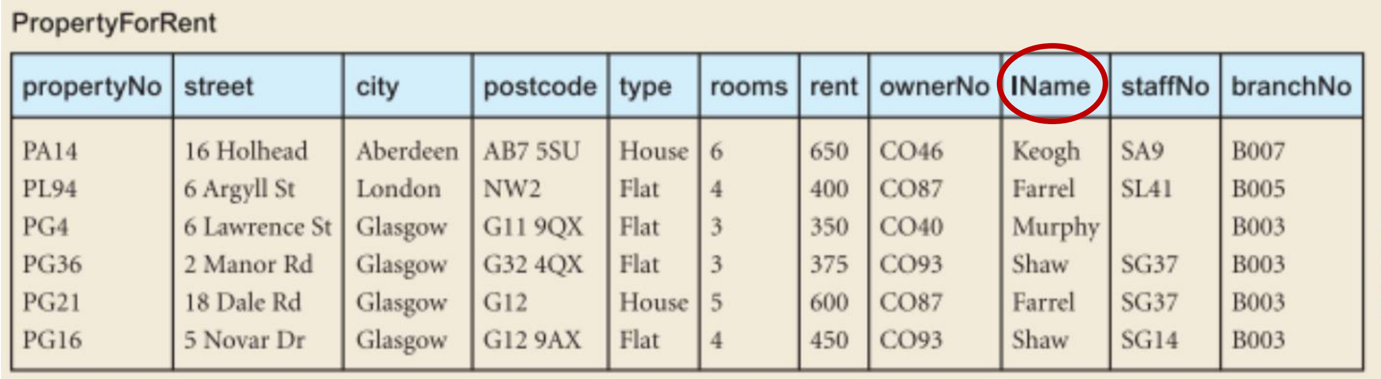
PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Staff

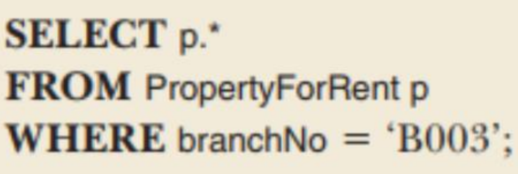
staffNo	fName	lName	branchNo	noOfProperties
SL21	John	White	B005	0
SG37	Ann	Beech	B003	2
SG14	David	Ford	B003	1
SA9	Mary	Howe	B007	1
SG5	Susan	Brand	B003	0
SL41	Julie	Lee	B005	1

Denormalization



PropertyForRent

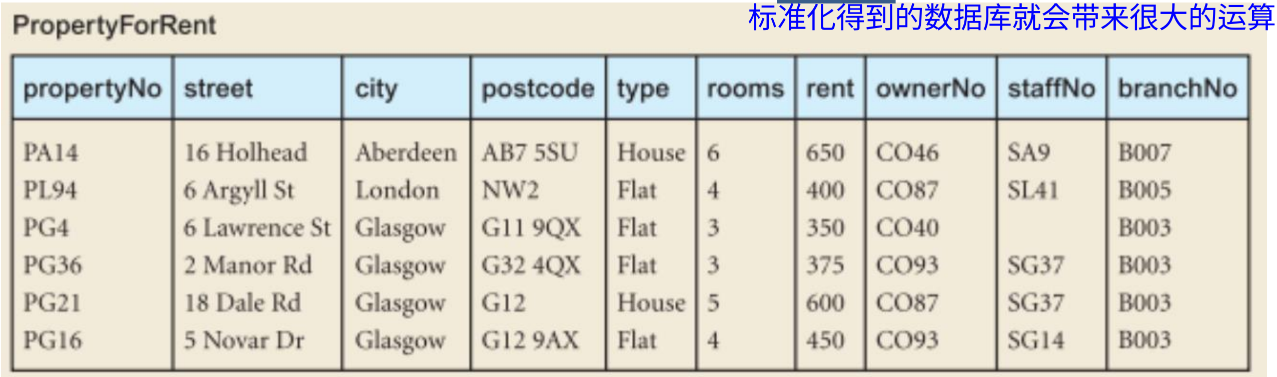
propertyNo	street	city	postcode	type	rooms	rent	ownerNo	lName	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	Keogh	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	Farrel	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	Murphy		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	Shaw	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	Farrel	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	Shaw	SG14	B003



```
SELECT p.*
FROM PropertyForRent p
WHERE branchNo = 'B003';
```

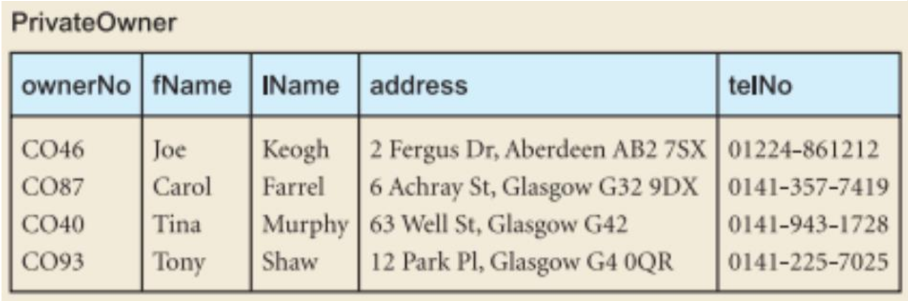
标准化最大的用处是“防止数据冗余”，但是有时候某个derived data经常要用到的时候

标准化得到的数据库就会带来很大的运算成本，这时候不标准化也挺好的



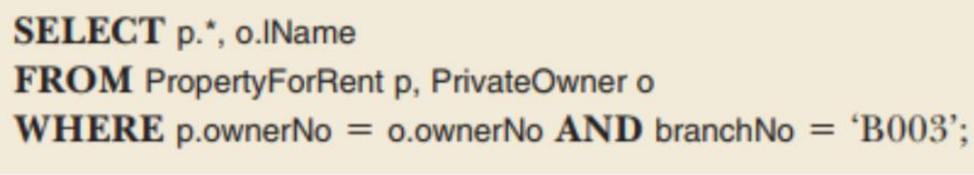
PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003



PrivateOwner

ownerNo	fName	lName	address	telNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025



```
SELECT p.*, o.lName
FROM PropertyForRent p, PrivateOwner o
WHERE p.ownerNo = o.ownerNo AND branchNo = 'B003';
```

Database Concepts (III)

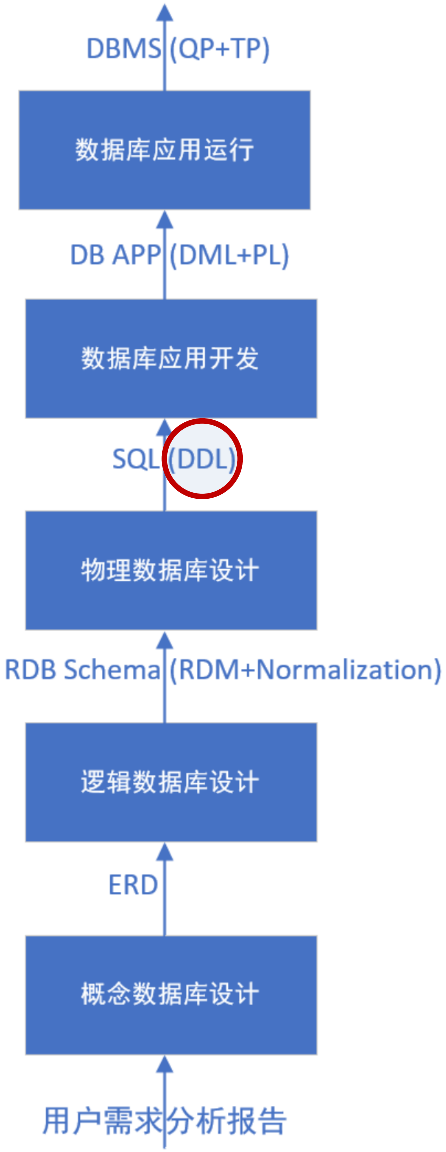
Structured Query Language

Chaokun Wang

School of Software, Tsinghua University
chaokun@tsinghua.edu.cn

April 11, 2022

Outline



DBMS (QP+TP)

数据库应用运行

DB APP (DML+PL)

数据库应用开发

SQL (DDL)

物理数据库设计

RDB Schema (RDM+Normalization)

逻辑数据库设计

ERD

概念数据库设计

用户需求分析报告

- Introduction to SQL
- Data Manipulation Language*
    - SELECT
    - INSERT
    - UPDATE
    - DELETE
- ✈ Data Definition Language*
    - Data Types
    - Schema
    - Table
    - Index
    - View
    - Transaction
- Procedural SQL

Data Definition

- A PG database cluster contains one or more named databases.
- A database contains one or more named database schemas.
- A database schema contain several kinds of named objects
  - data types, tables, views, indexes, functions, and operators

	Create	Change	Destroy
Schema	CREATE SCHEMA		DROP SCHEMA
Domain	CREATE DOMAIN	ALTER DOMAIN	DROP DOMAIN
Table	CREATE TABLE	ALTER TABLE	DROP TABLE
View	CREATE VIEW		DROP VIEW
Index	CREATE INDEX		DROP INDEX

SQL Identifiers

- Used to identify objects in the database, such as table names, view names, and columns.
- Characters must appear in a **character set**
  - ISO default character set consists of
    - The upper-case letters A . . . Z
    - The lower-case letters a . . . z
    - The digits 0 . . . 9, and
    - The underscore (_) character.
  - Restrictions
    - An identifier can be no longer than 128 characters (most dialects have a much lower limit than this);
    - An identifier must start with a letter;
    - An identifier cannot contain spaces.

Create/Destroy a Schema

cascade是在外码定义时指定的关键字。

外码所指定的字段取值受限制，可以取两种值：

所参照主码中出现过的值；可以取空值。

外码所指定的字段中数据的增删改是受到外码约束的限制的，在数据增删改时会检查是否满足外码约束条件，当不满足外码的条件时，所做的处理与定义外码时指定的restrict关键字或者cascade关键字有关。

- Create a schema
  - **CREATE SCHEMA** Name [**AUTHORIZATION** CreatorIdentifier]
    - **CREATE SCHEMA** SqlTests **AUTHORIZATION** Smith;
- Destroy a schema
  - **DROP SCHEMA** Name [**RESTRICT** | **CASCADE**]

限制条件

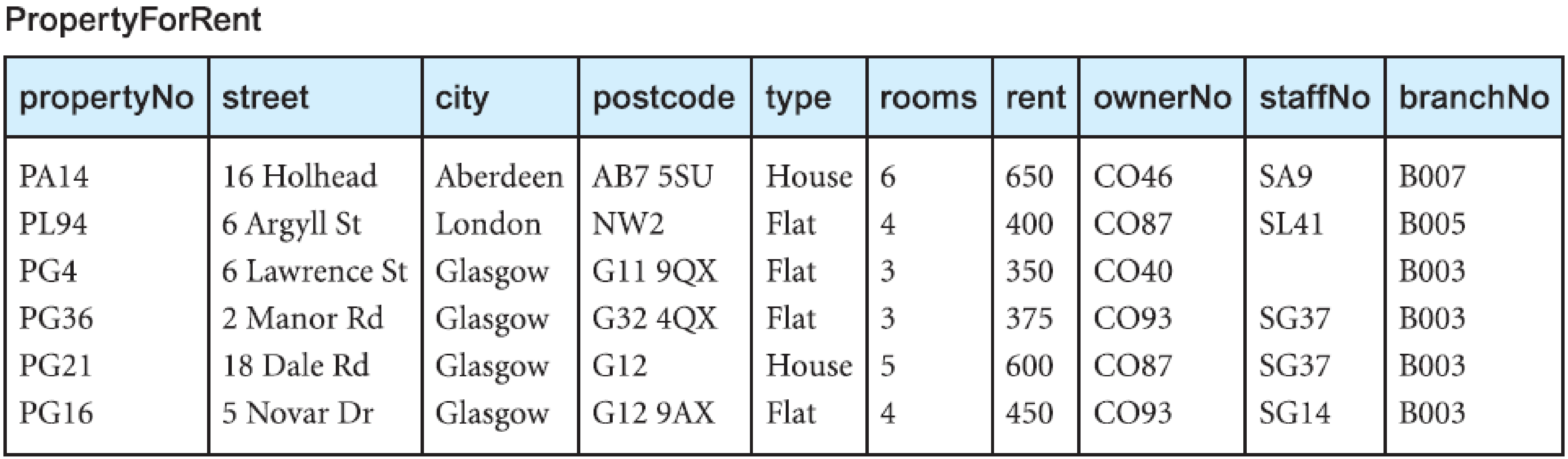
    - RESTRICT: the schema must be empty or the operation fails
    - CASCADE: the operation cascades to drop all objects associated with the schema

```
CREATE SCHEMA schema_name [AUTHORIZATION role_specification]
        [schema_element[ ... ]]
CREATE SCHEMA AUTHORIZATION role_specification [schema_element
        [...]]
CREATE SCHEMA IF NOT EXISTS schema_name [AUTHORIZATION
        role_specification]
CREATE SCHEMA IF NOT EXISTS AUTHORIZATION role_specification
```

```
DROP SCHEMA [IF EXISTS] name [, ...] [CASCADE | RESTRICT]
```

Create a Table

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)



PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

SQL: Create Table

```
CREATE TABLE PropertyForRent(
        propertyNo VARCHAR(5) NOT NULL,
        street VARCHAR(25) NOT NULL,
        city VARCHAR(15) NOT NULL,
        postcode VARCHAR(8),
        type CHAR(1) NOT NULL DEFAULT 'F' CHECK(type IN ('B', 'C', 'D', 'E', 'F', 'M', 'S')),
        rooms SMALLINT NOT NULL DEFAULT 4 CHECK(rooms BETWEEN 1 AND 15),
        rent DECIMAL(6,2) NOT NULL DEFAULT 600 CHECK(rent BETWEEN 0 AND 9999.99),
        ownerNo VARCHAR(5) NOT NULL,
        staffNo VARCHAR(5),
        branchNo CHAR(4) NOT NULL,
        PRIMARY KEY (propertyNo),
        FOREIGN KEY (staffNo) REFERENCES Staff ON DELETE SET NULL ON UPDATE CASCADE,
        FOREIGN KEY (ownerNo) REFERENCES PrivateOwner ON UPDATE CASCADE,
        FOREIGN KEY (branchNo) REFERENCES Branch ON UPDATE CASCADE
);
```

Name TYPE RESTRICT
比如说这里就规定了不能是空的

DEFAULT ‘F'意思是默认值设置为F

用CHECK来限制输入的值在这些里面

IN为关键字

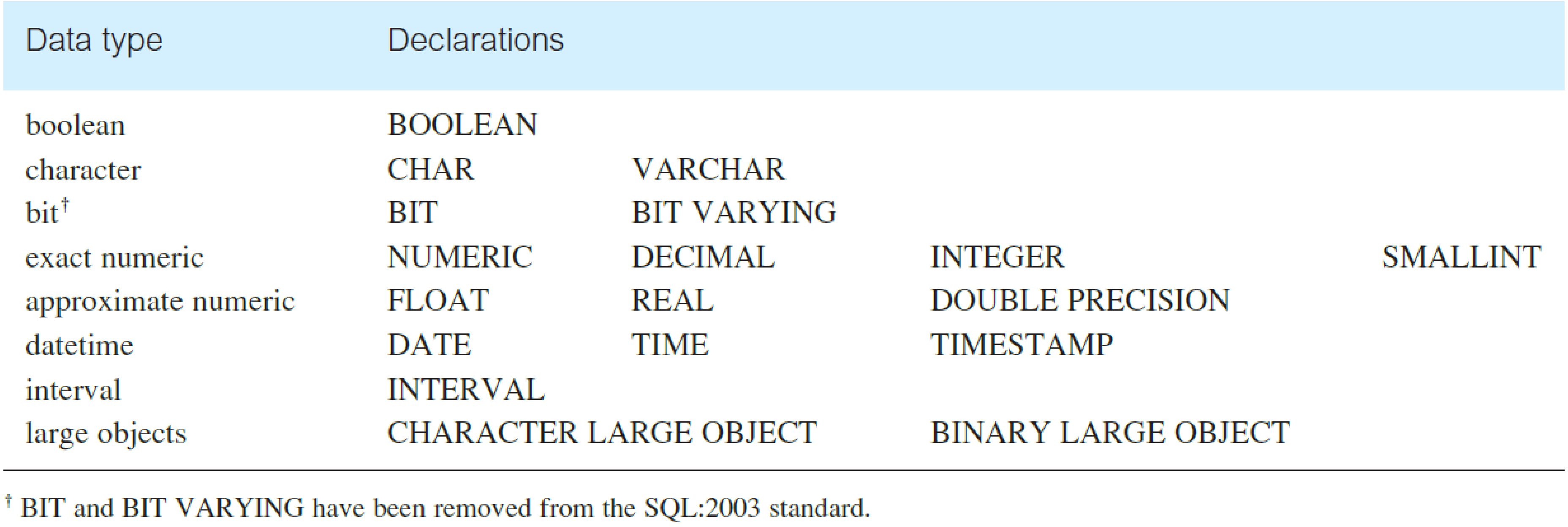
BETWEEN AND
为关键字

设置表格主键

这一句意思是，当ref表更新时，这个表里的数据怎么动

设置外键、外键来自的表格

Data Types



Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit[†]	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	
interval	INTERVAL			
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT	

[†] BIT and BIT VARYING have been removed from the SQL:2003 standard.

Data Types: Character

Data Type	SQL Server	Oracle	MySQL
Character	CHAR(n)	CHAR(n)	CHAR(n)
	8000 Bytes	2000 Bytes	255 Characters
National Character	NCHAR(n)	NCHAR(n)	
	4000 Characters	4000 Bytes	
Character Varying	VARCHAR(n)	VARCHAR2(n)	VARCHAR(n)
	8000 Bytes	2000 Bytes	65535 Bytes
National Char Varying	NVARCHAR(n)	NVARCHAR2(n)	
	4000 Characters	4000 Bytes	
Text	TEXT	CLOB	[MEDIUM|LONG]TEXT
	2^31 – 1 (2GB)	128 TB	64 KB/16 MB/4 GB
National Text	NTEXT	NCLOB	
	2^30 – 1 Characters	128 TB	

Data Types: Character

In PostgreSQL:

Name	Description
character varying(n), varchar(n)	variable-length with limit
character(n), char(n)	fixed-length, blank padded
text	variable unlimited length

Data Types: Numeric Data

Data Type	Bytes	Minimum (S)	Maximum (S)	SQL Server	MySQL
TINYINT	1	-128	127	Unsigned	Signed, Unsigned
SMALLINT	2	-32768	32767	Signed	Signed, Unsigned
MEDIUMINT	3	-8388608	8388607	Signed	Signed, Unsigned
INT	4	-2147483648	2147483647	Signed	Signed, Unsigned
BIGINT	8	-9.2233E+18	9.2233E+18	Signed	Signed, Unsigned

Data Type	Bytes	SQL Server	Oracle	MySQL
REAL	4	REAL	BINARY_FLOAT	FLOAT
DOUBLE PRECISION	8	FLOAT	BINARY_DOUBLE	DOUBLE
FLOAT [precision]		FLOAT(p)	FLOAT(p)	

Data Type	SQL Server	Oracle	MySQL

DECIMAL(precision[,scale])
NUMERIC(precision[,scale])
	38 (17 Bytes)	38	65 (29 Bytes)

Data Types: Numeric Data

In PostgreSQL:

Name	Storage Size	Range
smallint	2 bytes	-32768 to +32767
integer	4 bytes	-2147483648 to +2147483647
bigint	8 bytes	-9223372036854775808 to +9223372036854775807
decimal	variable	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	6 decimal digits precision
double precision	8 bytes	15 decimal digits precision
smallserial	2 bytes	1 to 32767
serial	4 bytes	1 to 2147483647
bigserial	8 bytes	1 to 9223372036854775807

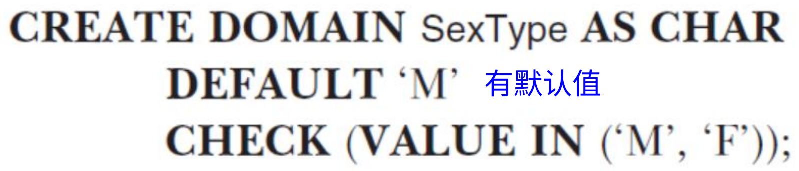
Data Types: Date & Time

Data Type	Fields	Example
DATE	YEAR, MONTH, DAY	'2014-11-04'
TIME [timePrecision]	HOUR, MINUTE, SECOND	'10:12:09.019473'
TIMESTAMP [timePrecision]	YEAR, MONTH, DAY, HOUR, MINUTE, SECOND	'2014-11-04 10:12:09.019473'
TIME [p] WITH TIME ZONE	…, TIMEZONE_HOUR, TIMEZONE_MINUTE	'10:12:09.019473 +08:00'
TIMESTAMP [p] WITH TIME ZONE	…, TIMEZONE_HOUR, TIMEZONE_MINUTE	'2014-11-04 10:12:09.019473 +08:00'

In PostgreSQL:

Name	Low Value	High Value	Resolution
timestamp [(p)] [without time zone]	4713 BC	294276 AD	1 microsecond
timestamp [(p)] with time zone	4713 BC	294276 AD	1 microsecond
date	4713 BC	5874897 AD	1 day
time [(p)] [without time zone]	00:00:00	24:00:00	1 microsecond
time [(p)] with time zone	00:00:00+1459	24:00:00-1459	1 microsecond
interval [fields] [(p)]	-178000000 years	178000000 years	1 microsecond

Create Domain

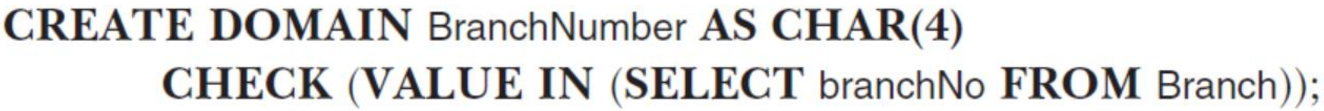


```
CREATE DOMAIN SexType AS CHAR
        DEFAULT 'M'
        CHECK (VALUE IN ('M', 'F'));
```

有默认值

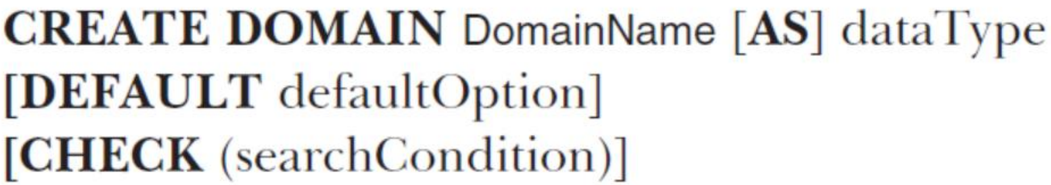
有约束

CREATE DOMAIN
它创建了一个用户定义的数据类型，可以有可选的约束。



```
CREATE DOMAIN BranchNumber AS CHAR(4)
        CHECK (VALUE IN (SELECT branchNo FROM Branch));
```

- The ISO standard allows domains to be defined more explicitly using the **CREATE DOMAIN** statement:



```
CREATE DOMAIN DomainName [AS] dataType
[DEFAULT defaultOption]
[CHECK (searchCondition)]
```

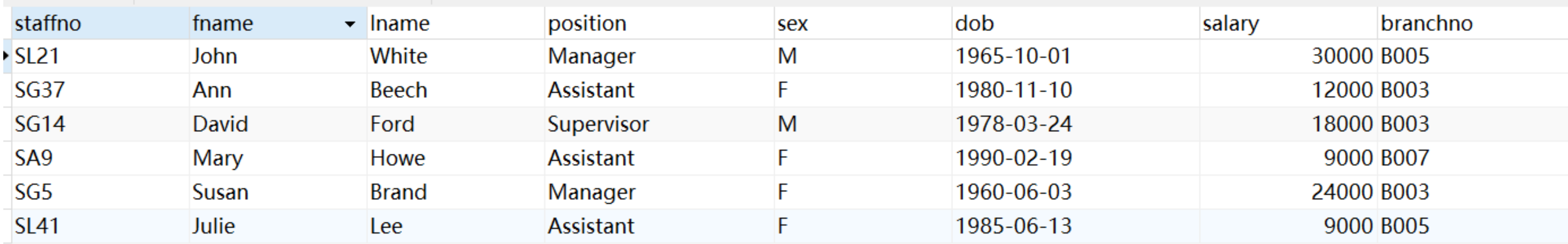
Integrity

- Required Data
  - position **VARCHAR**(10) **NOT NULL**
- Domain Constraints
  - sex **CHAR NOT NULL CHECK** (sex **IN** (‘M’, ‘F’))
- General Constraints
  - **CHECK** (**NOT EXISTS**
    (**SELECT** staffNo
    FROM PropertyForRent
    GROUP BY staffNo
    HAVING COUNT(*) > 100))

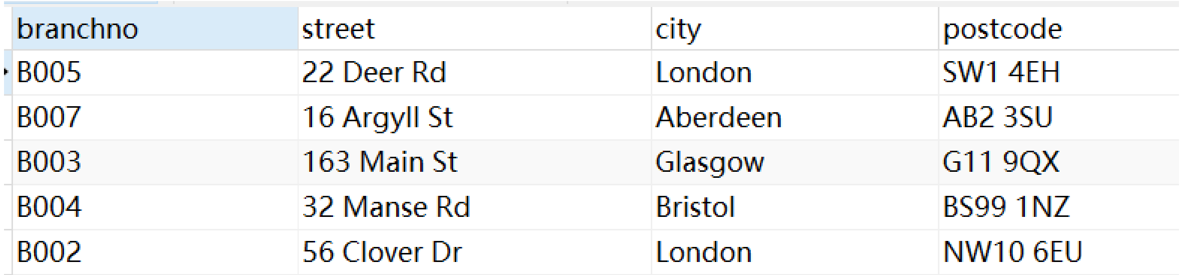
Integrity

- Entity Integrity
  - **PRIMARY KEY**(clientNo, propertyNo)
- Referential Integrity
  - **FOREIGN KEY**(branchNo) **REFERENCES** Branch
  - **FOREIGN KEY** (staffNo) **REFERENCES** Staff **ON DELETE SET NULL**
  - **FOREIGN KEY** (ownerNo) **REFERENCES** PrivateOwner **ON UPDATE CASCADE**
  - ON DELETE/UPDATE options
    - **CASCADE**: delete/update the matching rows in the child table
    - **SET NULL**: set the foreign key value(s) in the child table to NULL
    - **SET DEFAULT**: set the foreign key value(s) in the child table to the specified default value
    - **NO ACTION**: Reject the delete/update operation from the parent table 就是，我不让父表数据变动 的意思

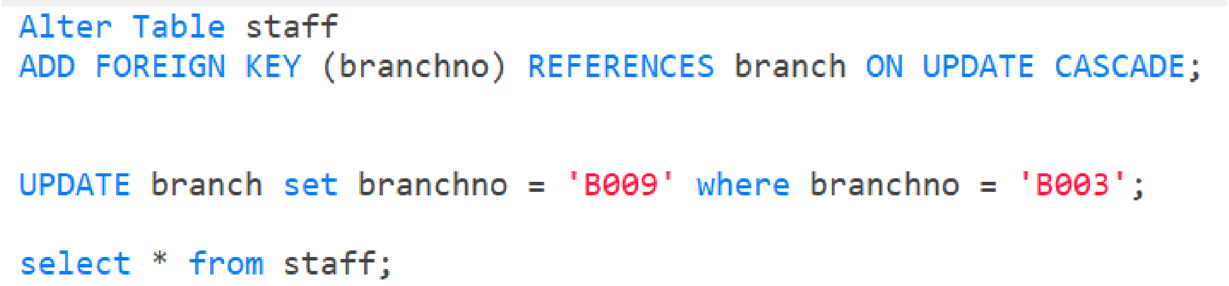
ON UPDATE CASCADE



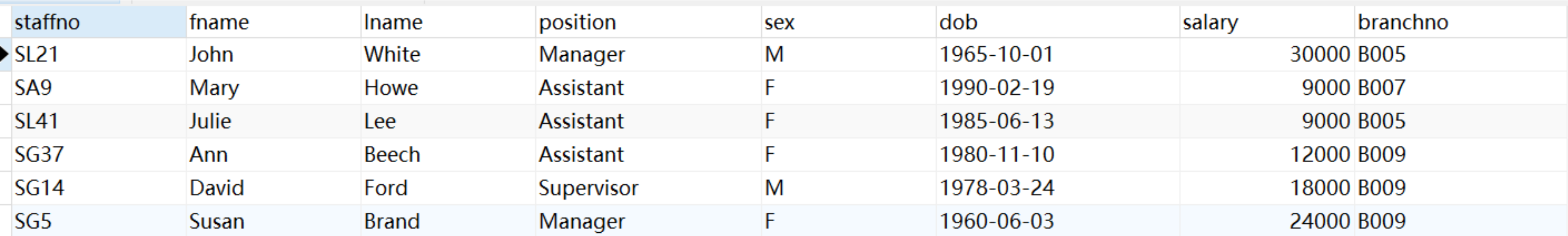
staffno	fname	lname	position	sex	dob	salary	branchno
SL21	John	White	Manager	M	1965-10-01	30000	B005
SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005



branchno	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

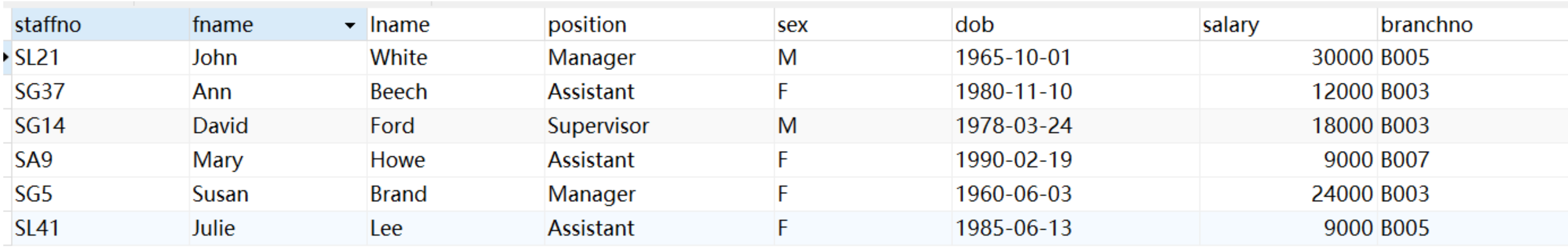


```
Alter Table staff
ADD FOREIGN KEY (branchno) REFERENCES branch ON UPDATE CASCADE;


UPDATE branch set branchno = 'B009' where branchno = 'B003';

select * from staff;
```

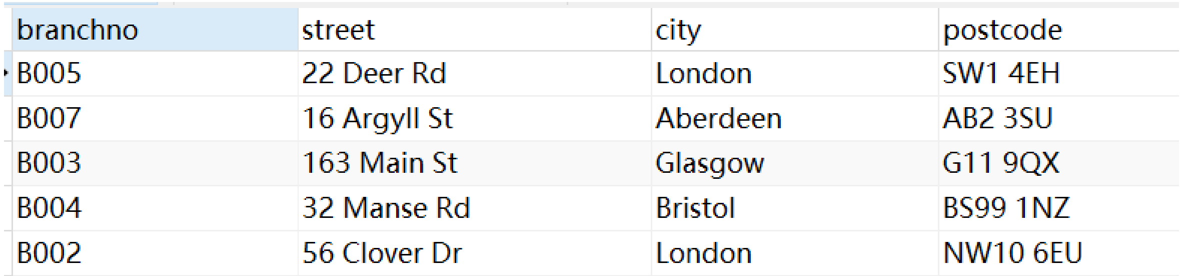


staffno	fname	lname	position	sex	dob	salary	branchno
SL21	John	White	Manager	M	1965-10-01	30000	B005
SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005
SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B009
SG14	David	Ford	Supervisor	M	1978-03-24	18000	B009
SG5	Susan	Brand	Manager	F	1960-06-03	24000	B009

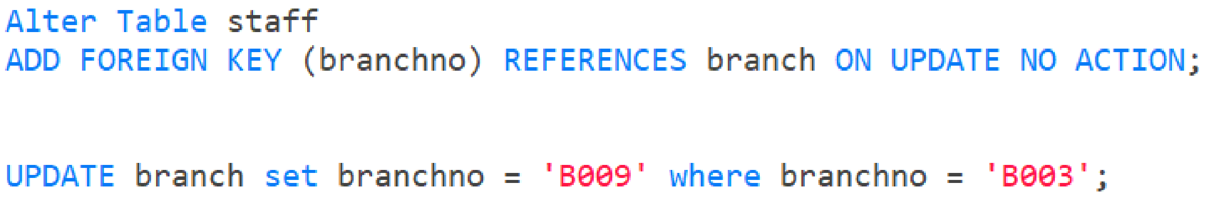
ON UPDATE NO ACTION



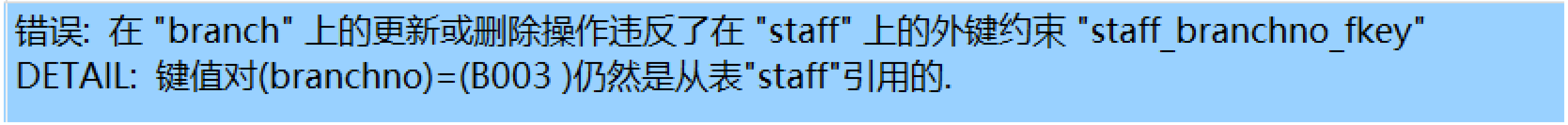
staffno	fname	lname	position	sex	dob	salary	branchno
SL21	John	White	Manager	M	1965-10-01	30000	B005
SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005



branchno	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

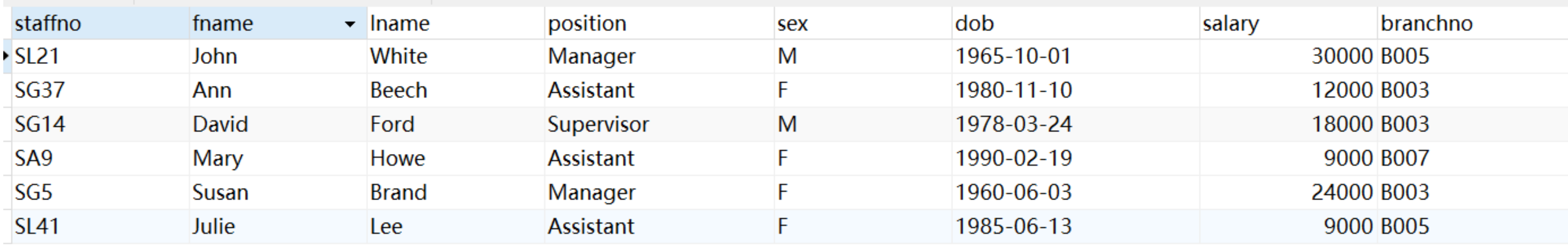


```
Alter Table staff
ADD FOREIGN KEY (branchno) REFERENCES branch ON UPDATE NO ACTION;


UPDATE branch set branchno = 'B009' where branchno = 'B003';
```

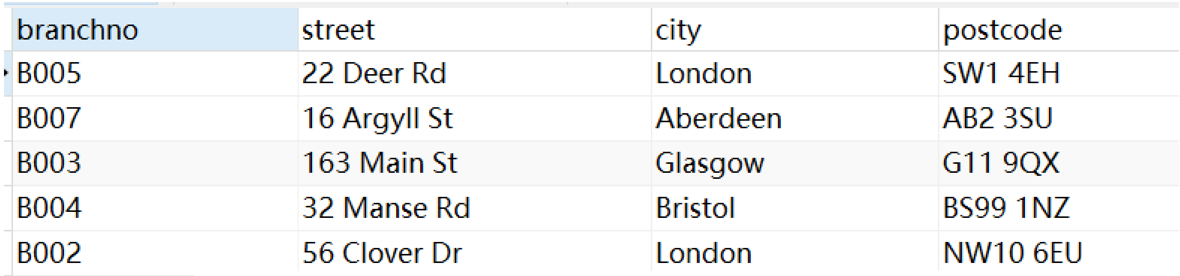


错误: 在 "branch" 上的更新或删除操作违反了在 "staff" 上的外键约束 "staff_branchno_fkey"
DETAIL: 键值对(branchno)=(B003)仍然是从表"staff"引用的.

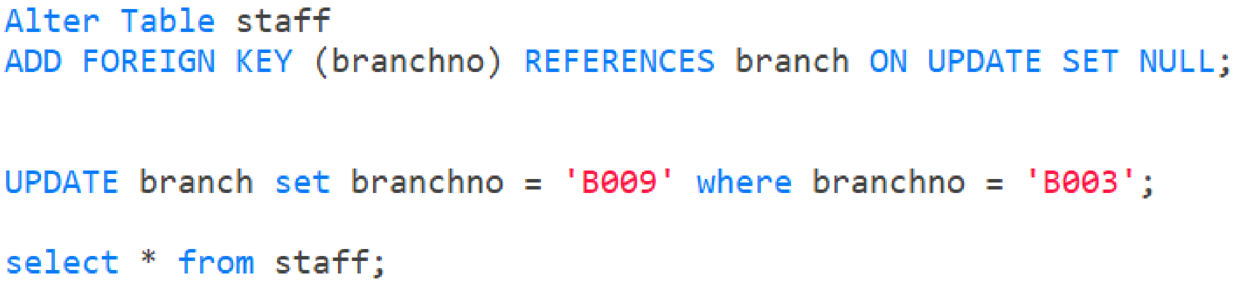
ON UPDATE SET NULL



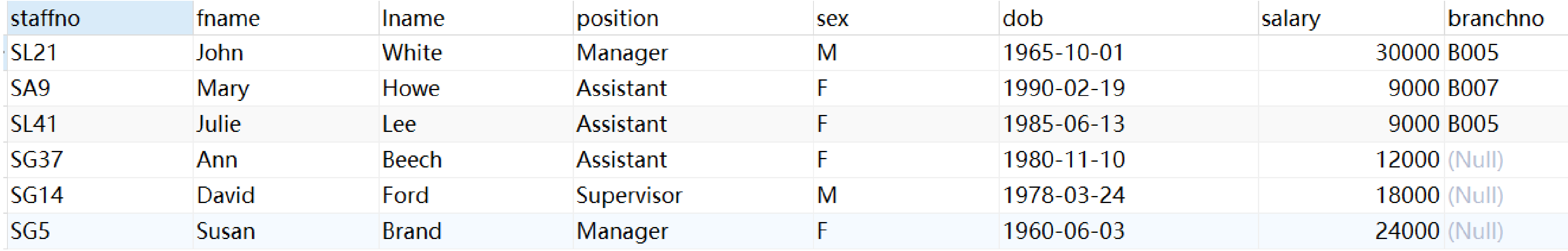
staffno	fname	lname	position	sex	dob	salary	branchno
SL21	John	White	Manager	M	1965-10-01	30000	B005
SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005



branchno	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

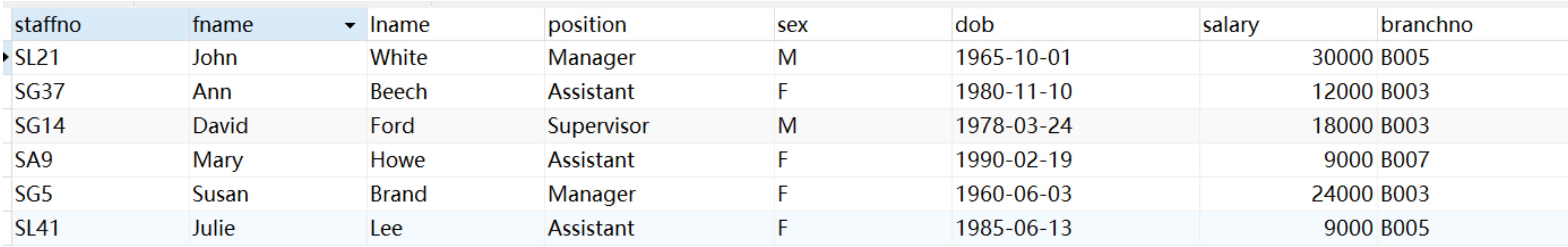


```
Alter Table staff
ADD FOREIGN KEY (branchno) REFERENCES branch ON UPDATE SET NULL;


UPDATE branch set branchno = 'B009' where branchno = 'B003';

select * from staff;
```

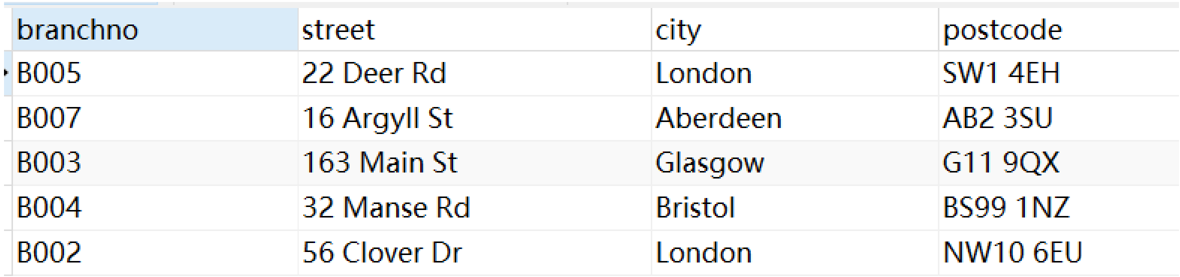


staffno	fname	lname	position	sex	dob	salary	branchno
SL21	John	White	Manager	M	1965-10-01	30000	B005
SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005
SG37	Ann	Beech	Assistant	F	1980-11-10	12000	(Null)
SG14	David	Ford	Supervisor	M	1978-03-24	18000	(Null)
SG5	Susan	Brand	Manager	F	1960-06-03	24000	(Null)

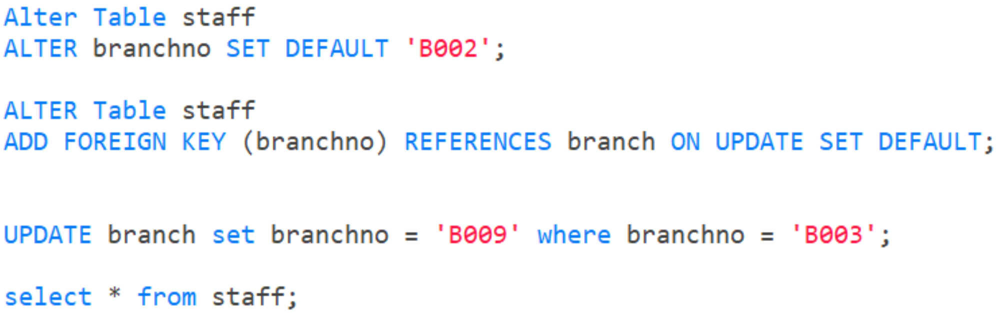
ON UPDATE SET DEFAULT



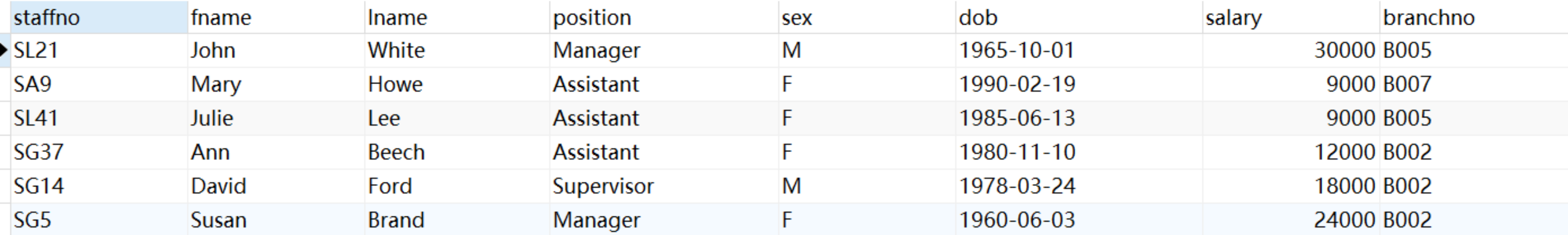
staffno	fname	lname	position	sex	dob	salary	branchno
SL21	John	White	Manager	M	1965-10-01	30000	B005
SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005



branchno	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU



```
Alter Table staff
ALTER branchno SET DEFAULT 'B002';

ALTER Table staff
ADD FOREIGN KEY (branchno) REFERENCES branch ON UPDATE SET DEFAULT;


UPDATE branch set branchno = 'B009' where branchno = 'B003';

select * from staff;
```



staffno	fname	lname	position	sex	dob	salary	branchno
SL21	John	White	Manager	M	1965-10-01	30000	B005
SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005
SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B002
SG14	David	Ford	Supervisor	M	1978-03-24	18000	B002
SG5	Susan	Brand	Manager	F	1960-06-03	24000	B002

SQL: Create/Destroy a Table

```
CREATE TABLE tbl_name (create_definition,...)

create_definition:
    col_name column_definition |
    PRIMARY KEY (col_name,...) |
    UNIQUE [index_name] (col_name,...) |
    INDEX [index_name] (col_name,...) |
    FOREIGN KEY [index_name] (col_name,...) reference_definition |
    CHECK (expr)

column_definition:
    data_type [NOT NULL] [DEFAULT default_value] [UNIQUE] [CHECK (expr)]

reference_definition:
    REFERENCES tbl_name (col_name,...) [MATCH {FULL|PARTIAL}]
    [ON DELETE reference_option] [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION
```

```
DROP TABLE [IF EXISTS] tbl_name [, ...] [RESTRICT | CASCADE]
```

SQL: Create Table

```
CREATE DOMAIN OwnerNumber AS VARCHAR(5)
        CHECK (VALUE IN (SELECT ownerNo FROM PrivateOwner));
CREATE DOMAIN StaffNumber AS VARCHAR(5)
        CHECK (VALUE IN (SELECT staffNo FROM Staff));
CREATE DOMAIN BranchNumber AS CHAR(4)
        CHECK (VALUE IN (SELECT branchNo FROM Branch));
CREATE DOMAIN PropertyNumber AS VARCHAR(5);
CREATE DOMAIN Street AS VARCHAR(25);
CREATE DOMAIN City AS VARCHAR(15);
CREATE DOMAIN Postcode AS VARCHAR(8);
CREATE DOMAIN PropertyType AS CHAR(1)
        CHECK(VALUE IN (‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘M’, ‘S’));
CREATE DOMAIN PropertyRooms AS SMALLINT;
        CHECK(VALUE BETWEEN 1 AND 15);
CREATE DOMAIN PropertyRent AS DECIMAL(6,2)
        CHECK(VALUE BETWEEN 0 AND 9999.99);
```

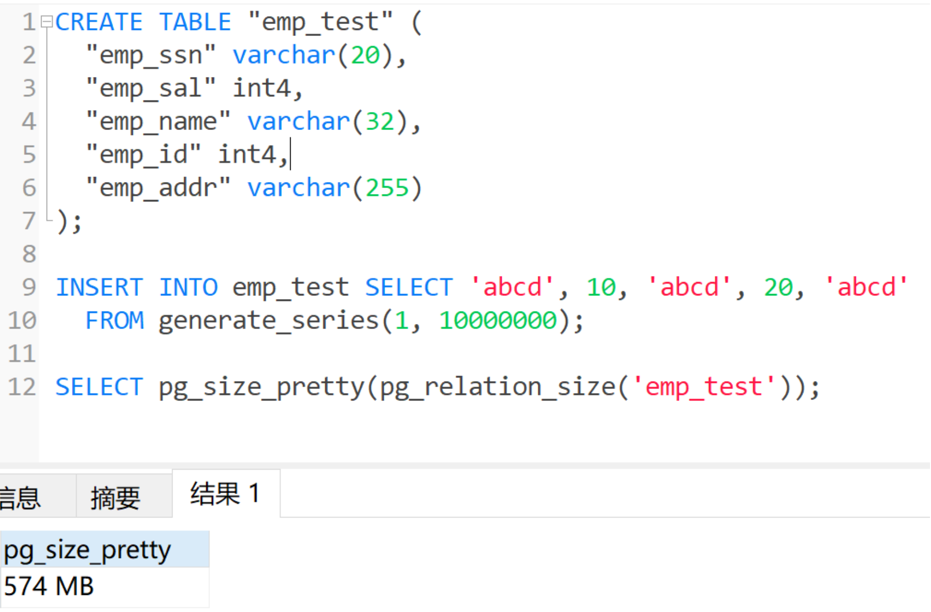
SQL: Create Table

```sql
CREATE TABLE PropertyForRent(
    propertyNo PropertyNumber NOT NULL,
    street Street NOT NULL,
    city City NOT NULL,
    postcode PostCode,
    type PropertyType NOT NULL DEFAULT 'F',
    rooms PropertyRooms NOT NULL DEFAULT 4,
    rent PropertyRent NOT NULL DEFAULT 600,
    ownerNo OwnerNumber NOT NULL,
    staffNo StaffNumber,
    branchNo BranchNumber NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff ON DELETE SET NULL ON UPDATE CASCADE,
    FOREIGN KEY (ownerNo) REFERENCES PrivateOwner ON UPDATE CASCADE,
    FOREIGN KEY (branchNo) REFERENCES Branch ON UPDATE CASCADE
);
```

全小写的是我们列名字，首字母大写是我们刚刚设定的Domain，
这样一来我们创建TABLE的时候就更加直观，代码也更短

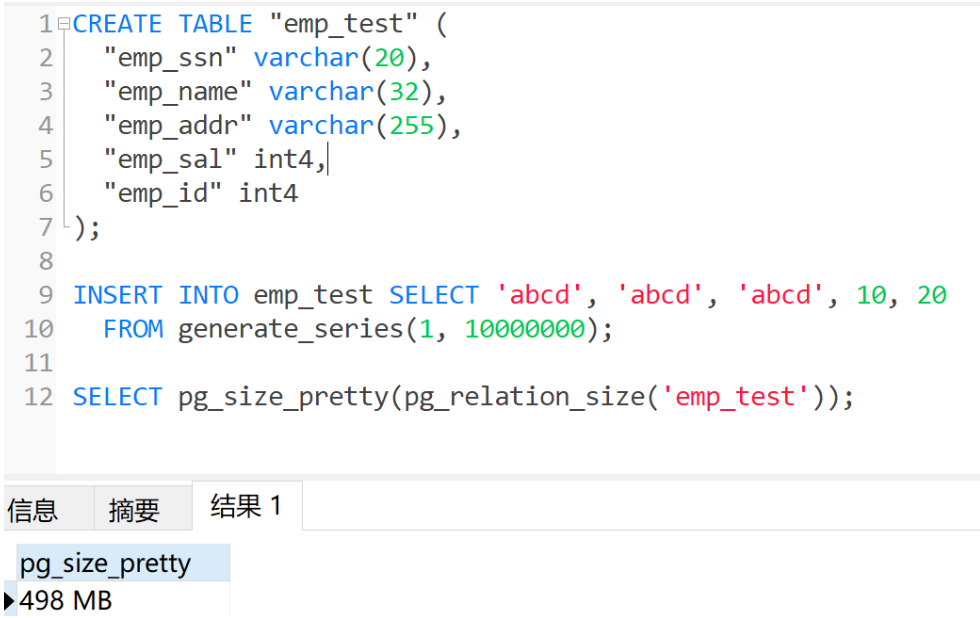
Grouping columns the right way

- Rethinking the effects of Physical Design



```
CREATE TABLE "emp_test" (
   "emp_ssn" varchar(20),
   "emp_sal" int4,
   "emp_name" varchar(32),
   "emp_id" int4,
   "emp_addr" varchar(255)
);

INSERT INTO emp_test SELECT 'abcd', 10, 'abcd', 20, 'abcd'
   FROM generate_series(1, 10000000);

SELECT pg_size_pretty(pg_relation_size('emp_test'));
```

```
pg_size_pretty
574 MB
```



```
CREATE TABLE "emp_test" (
   "emp_ssn" varchar(20),
   "emp_name" varchar(32),
   "emp_addr" varchar(255),
   "emp_sal" int4,
   "emp_id" int4
);

INSERT INTO emp_test SELECT 'abcd', 'abcd', 'abcd', 10, 20
   FROM generate_series(1, 10000000);

SELECT pg_size_pretty(pg_relation_size('emp_test'));
```

```
pg_size_pretty
498 MB
```

- PG aligns data physically.
- Group columns with similar data types next to each other.

Conclusions

- Database Normalization
  - Relation with FD
  - BCNF
  - Tutorial
  - Codd's Relational Database Rules
- Physical Database Design
  - Steps
  - Denormalization
- DDL
  - Data Types
  - Schema
  - Table

Homework

- Read Sections 8.1-8.4 of DS1
- Assignment
    - Later in Yuketang/Xuetang

Homework

- Read the following Chapters of DS1
  - § 8.1-8.2c (pp360-373)
  - § 9.7-9.8 (pp471-474)
  - § 3.9 (pp100-101)
- Assignment
  - Later in Xuetang
- Further Reading
  - § 18, § 19.1, § 7.1-7.3 of DS2

Thank you!