



# اصول طراحی پایگاه داده

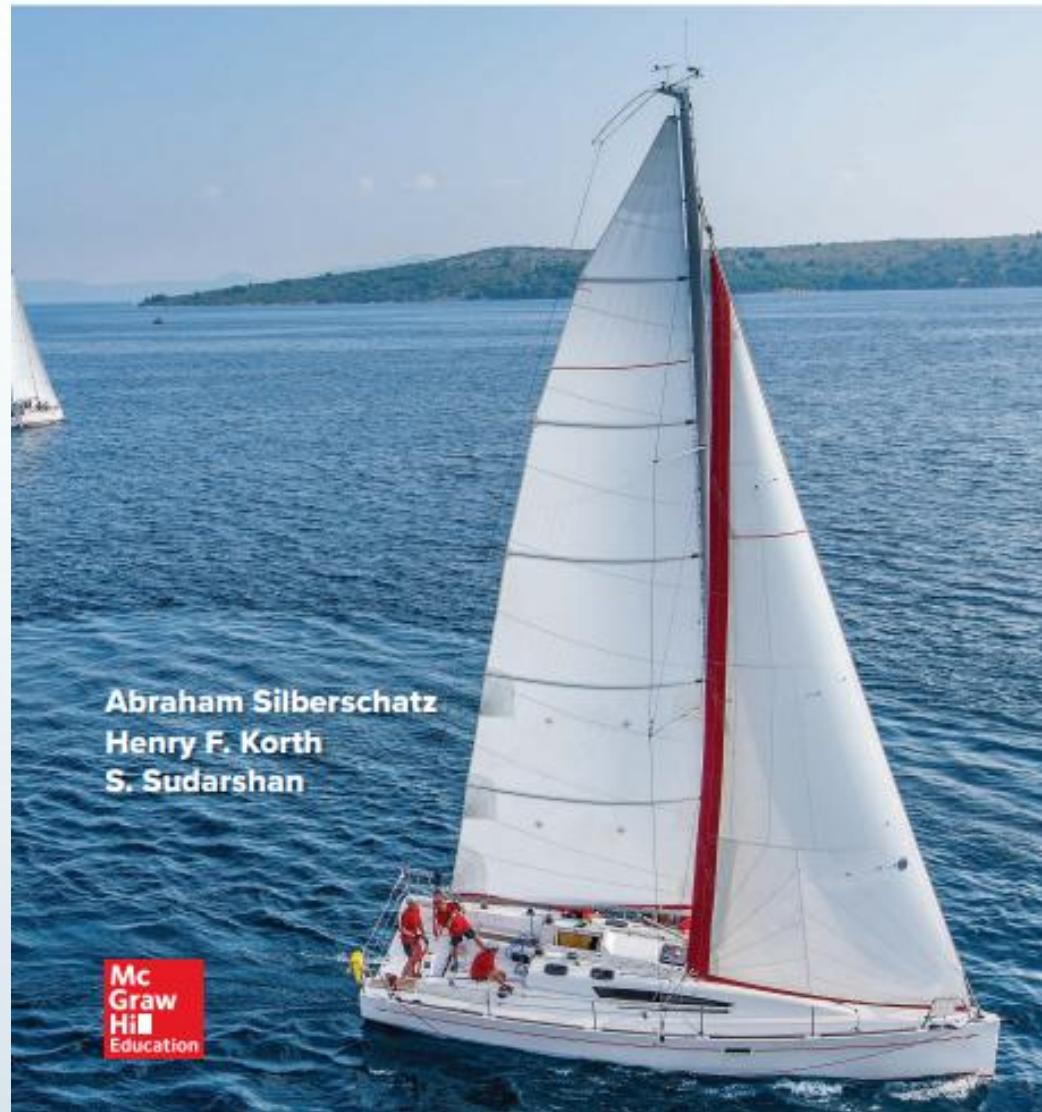
By Dr. Taghinezhad

Mail:

[a0taghinezhad@gmail.com](mailto:a0taghinezhad@gmail.com)

SEVENTH EDITION

## Database System Concepts





## فصل ۷: نرمال‌سازی

Database System Concepts, 7<sup>th</sup> Ed.

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# سرفصل‌ها

- ویژگی‌های یک طراحی رابطه‌ای خوب
- وابستگی‌های تابعی
- تجزیه جدول با استفاده از وابستگی‌های تابعی
- نرمال فرم‌ها
- تئوری وابستگی تابعی
- الگوریتم‌های تجزیه جدول با وابستگی‌های تابعی
- تجزیه با وابستگی‌های چندمقداری
- نرمال فرم‌های بیشتر
- دامنه‌های اتمی و نرمال فرم اول
- فرایند طراحی پایگاه داده
- مدل‌سازی داده‌های زمانی



# مروری بر نرمال‌سازی



# جداول آنرمال

## ■ نرمال سازی

- روشی برای طراحی جداول پایگاه داده و داده ها
- به طریقی که باعث کاهش افزونگی داده
- رفع مشکلات ساختاری و آنومالی



# جداول آنرمال

- هدف از نرمال سازی
- حذف افزونگی داده
- باقی نگاه داشتن وابستگی بین داده های مرتبط است.
- به این طریق اندازه پایگاه داده را کاهش داده و ذخیره منطقی داده را تضمین می کند.



مثال. جدول زیر که اطلاعات مربوط به خرید مشتریان را دارد در نظر بگیرید:

همانطور که مشاهده می شود با هر فروش داده ها در جدول تکرار می شوند.

Sale No	SaleDate	ProductNo	Qty	Amount	Salesrep	Customer No	First	Last	Address	CreditLimit
12345	Aug 12 2002	AQX88916	1	23.95	Dave Williams	4649-4673	Richard	Johnston	14 West Avenue	1000
12346	Aug 12 2002	AQX88916	7	167.65	Sara Thompson	1113-7741	Wayne	Jones	42 York Street	<null>
12347	Aug 13 2002	AHL46785	3705	5001.75	Li Qing	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12348	Aug 13 2002	DHU69863	50	118.5	Sara Thompson	<null>	<null>	<null>	<null>	<null>
12349	Aug 14 2002	DHU69863	940	2227.8	Sara Thompson	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12350	Aug 14 2002	DHU69863	42	99.54	Sara Thompson	7671-3496	Antonio	Gonzales	55B Granary Lane	<null>
12351	Aug 14 2002	AQX88916	55	1317.25	Dave Williams	6794-1674	Diane	Adams	364 East Road	150



## مثال. جدول زیر که اطلاعات مربوط به خرید مشتریان را دارد درنظر بگیرید:

این افزونگی مشکلات زیر را می تواند ایجاد کند:

■ هدر رفتن فضای ذخیره سازی.

- با وجودیکه امروزه دیسک های چند صد گیگابایتی وجود دارد چندین بار ذخیره یک داده غیر ضروری است.
- آنومالی در بهنگام سازی

- اگر داده یک مشتری، مثلاً آدرس، تغییر کند باید در همه جاهائی که ذخیره شده است این تغییر اعمال شود درغیراینصورت جامعیت نقص می شود.

■ آنومالی در حذف.

- اگر این جدول به منظور نگهداری مشخصات مشتریان باشد، اگر مشتری خریدش را پس بدهد و سطر مربوط به آن حذف شود کلیه اطلاعات مشتری هم حذف می شود.

■ آنومالی در درج.

- به همین صورت نمی توانیم مشخصات مشتری جدید را درج کنیم مگر اینکه کالائی خریده باشد.

Sale No	SaleDate	ProductNo	Qty	Amount	Salesrep	Customer No	First	Last	Address	CreditLimit
12345	Aug 12 2002	AQX88916	1	23.95	Dave Williams	4649-4673	Richard	Johnston	14 West Avenue	1000
12346	Aug 12 2002	AQX88916	7	167.65	Sara Thompson	1113-7741	Wayne	Jones	42 York Street	<null>
12347	Aug 13 2002	AHL46785	3705	5001.75	Li Qing	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12348	Aug 13 2002	DHU69863	50	118.5	Sara Thompson	<null>	<null>	<null>	<null>	<null>
12349	Aug 14 2002	DHU69863	940	2227.8	Sara Thompson	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12350	Aug 14 2002	DHU69863	42	99.54	Sara Thompson	7671-3496	Antonio	Gonzales	55B Granary Lane	<null>
12351	Aug 14 2002	AQX88916	55	1317.25	Dave Williams	6794-1674	Diane	Adams	364 East Road	150 8



- راهکار:
- جدا کردن داده های جدول زیر به جداول جداگانه افزونگی را کاهش می دهد
- مواجهه با آنومالی های فوق را ساده تر می کند.
- این فرآیند را نرمالسازی می نامند.

مثال. جدول زیر که اطلاعات مربوط به خرید مشتریان را دارد درنظر بگیرید:

Sale No	SaleDate	ProductNo	Qty	Amount	Salesrep	Customer No	First	Last	Address	CreditLimit
12345	Aug 12 2002	AQX88916	1	23.95	Dave Williams	4649-4673	Richard	Johnston	14 West Avenue	1000
12346	Aug 12 2002	AQX88916	7	167.65	Sara Thompson	1113-7741	Wayne	Jones	42 York Street	<null>
12347	Aug 13 2002	AHL46785	3705	5001.75	Li Qing	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12348	Aug 13 2002	DHU69863	50	118.5	Sara Thompson	<null>	<null>	<null>	<null>	<null>
12349	Aug 14 2002	DHU69863	940	2227.8	Sara Thompson	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12350	Aug 14 2002	DHU69863	42	99.54	Sara Thompson	7671-3496	Antonio	Gonzales	55B Granary Lane	<null>
12351	Aug 14 2002	AQX88916	55	1317.25	Dave Williams	6794-1674	Diane	Adams	364 East Road	150 9



- تئوری پایگاه داده درجه نرمالسازی جدول را با اصطلاح فرم های نرمال (**normal form**) شرح می دهد. فرم های نرمال (یا بطور خلاصه **NF**) معیاری برای تعیین درجه نرمال جدول در اختیار می گذارد.
- فرم های نرمال جداگانه روی هر جدول می توانند بکار بروند. پایگاه داده زمانی در فرم نرمال  $n$  خواهد بود که کل جداول آن در فرم نرمال  $n$  باشند.
- فرم های نرمال عبارتند از:

- - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Forth Normal Form (4NF)
  - Boyce/Codd Normal Form (BCNF)
  - Fifth Normal Form (5NF)
  - Domain/Key Normal Form (DKNF)



# جداول آنرمال

- جداول آنرمال به جداولی اطلاق میشود که در برخورد هر سطر با هر ستون آن به جای یک مقدار اتمی و تجزیه ناپذیر، مجموعه‌ای از مقادیر وجود دارد (مانند **Telephones**)

S#	Name	Telephones
7801	آرش	0311-6262778 0913-311-5234
7902	عسل	021-2956677 0912-314-4532



# جدول آنرمال

مهمترین عیب یک جدول آنرمال این است که برای هر یک از عملیات درج، حذف و اضافه به دو دسته عملیات درج تاپل و درج گروه اطلاعات مجموعه (Telephones) احتیاج است.

S#	Name	Telephones
7801	آرش	0311-6262778 0913-311-5234
7902	عسل	021-2956677 0912-314-4532



# جدول نرمال ۱

- یک جدول نرمال ۱ است اگر در برخورد هر سطر با هر ستون به یک مقدار تجزیه ناپذیر برسیم
- برای آنکه جدول آنرمال Student را به نرمال ۱ تبدیل کنیم، لازم است مقادیر ویژگیهای St# و Name را به ازاء هر شماره تلفن تکرار کنیم

S#	Name	Telephones
7801	آرش	0311-6262778
7801	آرش	0913-311-5234
7902	عسل	021-2956677
7902	عسل	0912-314-4532



## جدول نرمال ۱

■ مثال. جدول ALL\_SALES که اطلاعات فروش را نگهداری می کند درنظر بگیرید. ایا در فرم نرمال اول هست؟

Sale No	SaleDate	ProductNo	Qty	Amount	Salesrep	Customer No	First	Last	Address	CreditLimit
12345	Aug 12 2002	AQX88916	1	23.95	Dave Williams	4649-4673	Richard	Johnston	14 West Avenue	1000
12346	Aug 12 2002	AQX88916	7	167.65	Sara Thompson	1113-7741	Wayne	Jones	42 York Street	<null>
12347	Aug 13 2002	AHL46785	3705	5001.75	Li Qing	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12348	Aug 13 2002	DHU69863	50	118.5	Sara Thompson	<null>	<null>	<null>	<null>	<null>
12349	Aug 14 2002	DHU69863	940	2227.8	Sara Thompson	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12350	Aug 14 2002	DHU69863	42	99.54	Sara Thompson	7671-3496	Antonio	Gonzales	55B Granary Lane	<null>
12351	Aug 14 2002	AQX88916	55	1317.25	Dave Williams	6794-1674	Diane	Adams	364 East Road	150



## جدول نرمال ۱

■ این جدول در فرم اول نرمال هست چون هیچ کدام از ستون ها چندمقداری نیستند بنابراین نیازی نیست روی جدول کاری انجام دهیم بجز اینکه یک کلید انتخاب نمائیم.

Sale No	SaleDate	ProductNo	Qty	Amount	Salesrep	Customer No	First	Last	Address	CreditLimit
12345	Aug 12 2002	AQX88916	1	23.95	Dave Williams	4649-4673	Richard	Johnston	14 West Avenue	1000
12346	Aug 12 2002	AQX88916	7	167.65	Sara Thompson	1113-7741	Wayne	Jones	42 York Street	<null>
12347	Aug 13 2002	AHL46785	3705	5001.75	Li Qing	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12348	Aug 13 2002	DHU69863	50	118.5	Sara Thompson	<null>	<null>	<null>	<null>	<null>
12349	Aug 14 2002	DHU69863	940	2227.8	Sara Thompson	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12350	Aug 14 2002	DHU69863	42	99.54	Sara Thompson	7671-3496	Antonio	Gonzales	55B Granary Lane	<null>
12351	Aug 14 2002	AQX88916	55	1317.25	Dave Williams	6794-1674	Diane	Adams	364 East Road	150



## در پایگاه داده (Functional Dependency) وابستگی تابعی

- **وابستگی تابعی (FD)** صفت Y با صفت X وابستگی تابعی دارد اگر و فقط اگر در طول حیاط رابطه به هر مقدار X دقیقاً یک مقدار از Y متناظر باشد که می‌گوییم صفت X صفت Y را تعیین می‌کند.
- **تعریف:** اگر صفت X صفت Y را تعیین کند، گفته می‌شود Y به صورت تابعی وابسته به X است و به صورت زیر نمایش داده می‌شود:  
$$X \rightarrow Y$$
 ■  
در اینجا:  
■ تعیین‌کننده (Determinant) (نام دارد).  
■ وابسته (Dependent) (نام دارد).

شماره	نام	فamil
۱۱	اکبر	حسینی
۲۲	اکبر	کریمی



## در پایگاه داده (Functional Dependency) وابستگی تابعی

- FD1:  $\text{StudentID} \rightarrow \text{StudentName}$  با دانستن  $\text{StudentID}$  ، می‌توانیم به‌طور منحصر به‌فرد  $\text{StudentName}$  را تعیین کنیم.
- FD2:  $\text{StudentID} \rightarrow \text{Course}$  با دانستن  $\text{StudentID}$  ، می‌توانیم  $\text{Course}$  را تعیین کنیم.

StudentID	StudentName	Course
101	آلیس	ریاضی
102	باب	فیزیک
103	چارلی	شیمی

اگر  $B$  زیرمجموعه  $A$  باشد،  $A \rightarrow B$  وابستگی تابعی بدیهی می‌نمایم .

مثال:

$(\text{Sname}, \text{avg}) \rightarrow \text{avg}$



# ویژگی‌های یک طراحی رابطه‌ای خوب

- فرض کنید جدول `instructor` و `department` را یکی کنیم و جدولی به نام `in_dep` بسازیم.

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

این کار باعث تکرار اطلاعات می‌شود و در مواردی مجبور می‌شویم `NULL` وارد کنیم (مثلاً وقتی دپارتمانی استاد ندارد).



## در پایگاه داده (Functional Dependency) وابستگی تابعی

- وابستگی تابعی کامل FFD
- صفت خاصه  $y$  به صفت خاصه  $X$  وابستگی تابعی کامل دارد اگر  $y$  به  $X$  وابسته باشد ولی با هیچ یک از زیرمجموعه‌های  $X$  وابستگی تابعی نداشته باشد.
- رابطه دانشجوی زیر را در نظر بگیرید
- $S\#, SNAME, City, AVG, Department$
- وابستگی تابعی زیر وجود دارد
- $(S\#, SNAME) \rightarrow City$
- آیا وابستگی تابعی کامل هست؟
- اگر برای تمامی صفت‌های  $B$  داشته باشیم  $A \rightarrow B$  انگاه  $A$  ابر کلید هست.



## در پایگاه داده (Functional Dependency) وابستگی تابعی

- اگر  $F$  یک مجموعه از وابستگی های تابعی باشد آنگاه مجموعه تمام وابستگی های تابعی که از آن منتج می شود را مجموعه پوششی  $f^+$  نمایش می دهیم .



## قواعد استنتاج آرمسترانگ

قاعده انعکاسی

if  $B \subseteq A$  then  $A \rightarrow B \Rightarrow A \rightarrow A$  .1

تراگذاری یا قاعده تعددی

if  $A \rightarrow B$  and  $B \rightarrow C$  then  $A \rightarrow C$  .2

قاعده افزایش

if  $A \rightarrow B$  then  $(A, C) \rightarrow (B, C)$  .3

قاعده تجزیه if  $A \rightarrow (B, C)$  then  $A \rightarrow B$  and  $A \rightarrow C$  .4

قاعده ترکیب if  $A \rightarrow B$  and  $C \rightarrow D$  then  $(A, C) \rightarrow (B, D)$  .5

قاعده اجتماع

if  $A \rightarrow B$  and  $A \rightarrow C$  then  $A \rightarrow (B, C)$  .6

قاعده شبه تعددی

if  $A \rightarrow B$  and  $(B, C) \rightarrow D$  then  $(A, C) \rightarrow D$  .7



## کاربردهای قواعد آرمسترانگ

- ۱) محاسبه بستار صفت  $A^+ : A$  :

  - مجموعه تمام صفاتی که با  $A$ ، وابستگی تابعی دارند
  - نکته: اگر  $A^+ = H_R$  در این صورت  $A$  سوپر کلید (الگوریتم تشخیص سوپر کلید)

- ۲) محاسبه بستار مجموعه وابستگی‌های تابعی یک رابطه :  $F^+$

  - مجموعه تمام FD هایی که از  $F$  منطقاً استنتاج میشوند
  - :  $F = \{A \rightarrow B, B \rightarrow C\} \Rightarrow F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, (A, C) \rightarrow (B, C), \dots\}$



# Finding $F^+$

▪ اگر رابطه‌ی  $R(A,B,C,D)$  با وابستگی‌های تابعی  $F$  را داشته باشیم.

1.  $A \rightarrow B$

2.  $B \rightarrow C$

3.  $A \rightarrow D$

▪ پیدا کردن  $F^+$ -بستار مجموعه شامل تمام وابستگی‌های تابعی مشتق شده است.

1.  $A \rightarrow B$

2.  $B \rightarrow C$

3.  $A \rightarrow D$

4.  $A \rightarrow CA$  (تعدی)

5.  $A \rightarrow BCA$  (افزایش)

6.  $A \rightarrow BCDA$  (ترکیب وابستگی)



## (2NF) Second Normal Form

### ▪ ۲NF تعریف

۱. جدول باید ابتدا در ۱NF باشد

۲. تمام صفات غیر کلیدی باید به طور کامل به کل کلید اصلی وابسته باشند (نه فقط بخشی از کلید اصلی). اگر وابستگی جزئی وجود داشته باشد، جدول به ۲NF نمی‌رسد.

ستون  $Y$  با ستون  $X$  در یک رابطه وابستگی تابعی (functional dependency) دارد اگر و فقط اگر به ازای هر مقدار در  $X$  دقیقاً یک مقدار در  $Y$  متناظر با آن وجود داشته باشد. که به صورت  $Y \rightarrow X$  نشان داده می‌شود.



## جدول نرمال ۲

آیا جدول زیر 2NF هست؟

Sale No	SaleDate	Product No	Qty	Amount	Salesrep	CustomerNo	First	Last	Address	CreditLimit
12345	Aug 12 2002	AQX88916	1	23.95	Dave Williams	4649-4673	Richard	Johnston	14 West Avenue	1000
12346	Aug 12 2002	AQX88916	7	167.65	Sara Thompson	1113-7741	Wayne	Jones	42 York Street	<null>
12347	Aug 13 2002	AHL46785	3705	5001.75	Li Qing	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12348	Aug 13 2002	DHU69863	50	118.5	Sara Thompson	<null>	<null>	<null>	<null>	<null>
12349	Aug 14 2002	DHU69863	940	2227.8	Sara Thompson	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12350	Aug 14 2002	DHU69863	42	99.54	Sara Thompson	7671-3496	Antonio	Gonzales	55B Granary Lane	<null>
12351	Aug 14 2002	AQX88916	55	1317.25	Dave Williams	6794-1674	Diane	Adams	364 East Road	150



## یک شِمای ترکیبی بدون تکرار

همه ترکیب‌ها الزاماً باعث تکرار اطلاعات نمی‌شوند.

■ در نظر بگیرید جدول section و sec\_class را ترکیب کنیم

- *sec\_class(sec\_id, building, room\_number)* and
- *section(course\_id, sec\_id, semester, year)*

into one relation

- *section(course\_id, sec\_id, semester, year,  
building, room\_number)*

■ در این حالت تکراری رخ نمی‌دهد



## جدول نرمال ۲

- با کلید اصلی ترکیبی (SaleNo, ProductNo, CustomerNo)، وابستگی صفات غیر کلیدی را بررسی می کنیم:

## Credit Limit: Address .Last .First .1

این صفات فقط به **CustomerNo** وابسته هستند و به کل کلید ترکیبی **(SaleNo, ProductNo, CustomerNo)** وابسته نیستند.

○ این موضوع باعث ایجاد وابستگی جزئی می‌شود و فرم ۲ NF را نقض می‌کند.

2. فقط به SaleNo وابسته است (با فرض اینکه هر فروش نماینده فروش خاصی دارد) و به کل کلید ترکیبی Salesrep: (SaleNo, ProductNo, CustomerNo) وابسته نیست.

اين نيز يك وابستگي جزئي است که فرم ۲NF را نقض می کند.

نقط نمی کند.

Sale No	Sale Date	Product No	Qty	Amount	Salesrep	Tarife	Customer No	First Name	Last Name	Address	Credit Limit
	NFT	می گوئیم جدول است.									
12345	Aug 12 2002	AQX88916	1	23.95	Dave Williams		4649-4673	Richard	Johnston	14 West Avenue	1000
12346	Aug 12 2002	AQX88916	7	167.65	Sara Thompson		1113-7741	Wayne	Jones	42 York Street	<null>
12347	Aug 13 2002	AHL46785	3705	5001.75	Li Qing		1166-3461	Amelia	Waverley	995 Forth Street	<null>
12348	Aug 13 2002	DHU69863	50	118.5	Sara Thompson		<null>	<null>	<null>	<null>	<null>
12349	Aug 14 2002	DHU69863	940	2227.8	Sara Thompson		1166-3461	Amelia	Waverley	995 Forth Street	<null>
12350	Aug 14 2002	DHU69863	42	99.54	Sara Thompson		7671-3496	Antonio	Gonzales	55B Granary Lane	<null>
Dr. A. Taghinezhad	Aug 14 2002	AQX88916	55	1317.25	Dave Williams	1.27	6794-1674	Diane	Adams	364 East Road	150 Database



مثال. جدول ALL\_SALES را در نظر بگیرید:

ALL\_SALES(SaleNo, ProductNo, CustomerNo, SaleDate, QtyInStock, Description, Price, Customer\_Name, CreditLimit, Amount, Salesrep)

مشاهده می شود بعضی از ستون ها بهم مرتبط هستند و توسط بخشی از کلید مشخص می شوند.  
به عبارت دیگر بعضی ستون ها با زیرمجموعه ای از کلید وابستگی تابعی دارند:

ProductNo → {Description, ReorderLevel, Price, QtyInStock}

CustomerNo → {Customer\_Name, CreditLimit}

SaleNo → {Date, CustomerNo, ProductNo, Qty, Amount, Salesrep}

Sale No	SaleDate	ProductNo	Qty	Amount	Salesrep	Customer No	First	Last	Address	CreditLimit
12345	Aug 12 2002	AQX88916	1	23.95	Dave Williams	4649-4673	Richard	Johnston	14 West Avenue	1000
12346	Aug 12 2002	AQX88916	7	167.65	Sara Thompson	1113-7741	Wayne	Jones	42 York Street	<null>
12347	Aug 13 2002	AHL46785	3705	5001.75	Li Qing	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12348	Aug 13 2002	DHU69863	50	118.5	Sara Thompson	<null>	<null>	<null>	<null>	<null>
12349	Aug 14 2002	DHU69863	940	2227.8	Sara Thompson	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12350	Aug 14 2002	DHU69863	42	99.54	Sara Thompson	7671-3496	Antonio	Gonzales	55B Granary Lane	<null>
12351	Aug 14 2002	AQX88916	55	1317.25	Dave Williams	6794-1674	Diane	Adams	364 East Road	150



با جدا کردن این ستون ها به جداول جداگانه به فرم دوم نرم‌مال می‌رسیم.

- PRODUCT(ProductNo, Description, Price, QtyInStock)
- CUSTOMER(CustomerNo, Customer\_Name, CreditLimit)
- SALE(SaleNo, Date, CustomerNo, ProductNo, Qty, Amount, Salesrep)

Sale No	SaleDate	ProductNo	Qty	Amount	Salesrep	Customer No	First	Last	Address	CreditLimit
12345	Aug 12 2002	AQX88916	1	23.95	Dave Williams	4649-4673	Richard	Johnston	14 West Avenue	1000
12346	Aug 12 2002	AQX88916	7	167.65	Sara Thompson	1113-7741	Wayne	Jones	42 York Street	<null>
12347	Aug 13 2002	AHL46785	3705	5001.75	Li Qing	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12348	Aug 13 2002	DHU69863	50	118.5	Sara Thompson	<null>	<null>	<null>	<null>	<null>
12349	Aug 14 2002	DHU69863	940	2227.8	Sara Thompson	1166-3461	Amelia	Waverley	995 Forth Street	<null>
12350	Aug 14 2002	DHU69863	42	99.54	Sara Thompson	7671-3496	Antonio	Gonzales	55B Granary Lane	<null>
12351	Aug 14 2002	AQX88916	55	1317.25	Dave Williams	6794-1674	Diane	Adams	364 East Road	150



# جدول نرمال ۲

■ یک جدول نرمال ۲ است اگر:

- نرمال ۱ باشد

- در آن هیچ وابستگی جزئی به کلید اصلی وجود نداشته باشد. به عبارت دیگر، هیچ ویژگی جدول تنها به قسمتی از کلید اصلی وابستگی نداشته باشد.

S#	Name	Crs#	Cname	Unit	Grade	Term
7801	علی	1400	پایگاه داده	3	20	79-2
7801	علی	1500	ریاضی ۱	3	10	80-1
7801	علی	1600	تجزیه و تحلیل	3	20	80-1
7902	عسل	1400	پایگاه داده	3	7	80-1
7902	عسل	1700	تربیت بدنی	1	20	80-1

بخشی از جدول فوق وابسته به Crs# است و بخش دیگر وابسته به S# است



# جدول نرمال ۳

- یک جدول در فرم سوم نرمال است اگر او لا  $3NF$  باشد، ثانیا کلیه صفات خاصه غیر کلید در جدول با کلید اصلی وابستگی تابعی غیر تعدی داشته باشند.
- وابستگی تعدی **transitive dependency** یک وابستگی تابعی غیر مستقیم است که در آن  $Z \rightarrow X \rightarrow Y$  است اگر  $Z \rightarrow Y$  باشد.
- در فرم سوم نرمال کلیه ستون های جدول مستقیماً توسط کلید اصلی مشخص می شوند.
- با حذف فیلد هائی که وابستگی مستقیم با کلید ندارند به فرم سوم نرمال می رسیم. برای این کار گروهی از ستون های جدول را که مقدارشان برای بیش از یک رکورد تکرار می شود را در جدول جداگانه ای قرار دهید.



# جدول نرمال ۳

یک جدول نرمال ۳ است اگر:

- نرمال ۲ باشد
- در آن هیچ وابستگی تعددی (وابستگی با واسطه) در آن وجود نداشته باشد. به عبارت دیگر، در آن هیچ ویژگی غیر کلیدی به ویژگی غیر کلیدی دیگر وابستگی تابعی نداشته باشد.

Prof#	Pname	LastDegree	LastDegreeName
7801	علی	2	کاردانی
7802	آرش	2	کاردانی
7803	عسل	4	فوق لیسانس

برای آنکه این جدول نرمال ۳ شود به صورت زیر تبدیل میشود:

- Prof(Prof#, Pname, LastDegree)
- Degree(LastDegree#, LastDegreeName)



# جدول نرمال ۳

- مثال. فرض کنید جدول PRODUCT به صورت زیر جزئیات تولید کننده هر محصول را دارا باشد:  
**PRODUCT(ProductNo, Description, ReorderLevel, Price, QtyInStock, SupplierCode, SupplierName, SupplierAddress)**

این جدول کلید اصلی تک ستونی دارد بنابراین ۲NF است. اگر تولید کننده چندین محصول را تولید کند فیلدهای SupplierAddress و SupplierName برای هر محصول تکرار می شود زیرا وابستگی تعدی با کلید اصلی دارد.

**ProductNo → SupplierCode → {SupplierName, SupplierAddress}**

با حذف این ستون ها و تقسیم جدول به صورت زیر به فرم سوم نرمال می رسیم. توجه کنید که در جدول PRODUCT به عنوان کلید خارجی باقی می ماند.

**PRODUCT(ProductNo, Description, ReorderLevel, Price, QtyInStock, SupplierCode)**  
**SUPPLIER(SupplierCode, SupplierName, SupplierAddress)**



# جداول نرمال BCNF

- یک جدول نرمال BCNF است اگر و تنها اگر کلیه تعیین کننده های (Determinant) آن، کلید کاندیدا باشند. یعنی هر رابطه  $A \rightarrow B$  در جدول وجود داشته باشد کلید کاندیدا باشد.



# جداول نرمال BCNF

یکی از سطوح بالای نرمال‌سازی در **BCNF (Boyce-Codd Normal Form)** ■

پایگاه داده است که برای کاهش انحرافات و مشکلات ناشی از تکرار داده‌ها طراحی شده است .

در حقیقت یک شکل اصلاح شده از **3NF (Third Normal Form)** است که شرایط سختگیرانه‌تری را برای روابط در پایگاه داده وضع می‌کند.

## ■ شرایط BCNF:

■ یک رابطه (Table) در BCNF قرار دارد اگر برای هر وابستگی تابعی  $Y \rightarrow X$  در آن رابطه، یکی از شرایط زیر برقرار باشد:

1.  $X$  باید یک سوپرکلید باشد، یعنی مجموعه‌ای از ویژگی‌ها (attributes) که می‌تواند به طور منحصر به فرد هر سطر را شناسایی کند.

2. در غیر این صورت، باید وابستگی‌های غیرضروری (جزئی) وجود نداشته باشد.



# جداول نرمال BCNF

فرض کنید یک جدول به نام **Student\_Course** که اطلاعات مربوط به دانشجویان و دوره‌هایی که در آن‌ها ثبت‌نام کردند را ذخیره می‌کند: در این جدول، وابستگی‌های تابعی به شکل زیر هستند:

برای هر ترکیب منحصر به فرد از **Student\_ID, Course\_ID → Instructor**: .1 دانشجو و دوره، استاد مشخص است.

هر دوره یک استاد ثابت دارد. **Course\_ID → Instructor**: .2

مشکل این است که در این جدول، **Course\_ID → Instructor** نشان‌دهنده یک وابستگی است که باید در BCNF تجزیه شود زیرا **Course\_ID** یک سوپرکلید نیست (این تنها بخشی از کلید ترکیبی است).

Student_ID	Course_ID	Instructor	Semester
1	101	Dr. A	Fall 2024
2	102	Dr. B	Spring 2024
3	101	Dr. A	Fall 2024



# تبديل به جدول نرمال BCNF

1. جدول Course که اطلاعات دوره‌ها و استادها را ذخیره می‌کند:
2. جدول Student\_Course که اطلاعات مربوط به دانشجویان و دوره‌ها را نگهداری می‌کند:  
در این حالت، هر جدول به BCNF رسیده است، زیرا در هیچ‌کدام از جداول،  
وابستگی تابعی وجود ندارد که ویژگی‌های چپ آن‌ها سوپرکلید نباشد.

Course_ID	Instructor
101	Dr. A
102	Dr. B

Student_ID	Course_ID	Semester
1	101	Fall 2024
2	102	Spring 2024
3	101	Fall 2024



# جداول نرمال BCNF

■ آیا BCNF هست ؟

در جدول زیر دو کلید کاندیدای S#+Tutor و S#+Field و نیز داریم بنابراین میتواند به دو جدول تقسیم شود

S#	Field	Tutor
۷۸۰۱	مهندس کامپیوتر	مهندس رضائی
۷۸۰۱	ریاضی محض	آرش ریاضیدان
۷۸۰۱	هنر	گلناز هنردوست
۷۹۰۲	مهندس کامپیوتر	مجید رضائی
۷۸۰۳	مهندس کامپیوتر	پروین صبا



# تجزیه (Decomposition)

■ همه تجزیه‌ها خوب نیستند. فرض کنیم که ما جدول را این‌گونه تجزیه کنیم:

*employee*(*ID*, *name*, *street*, *city*, *salary*)

into

*employee1* (*ID*, *name*)

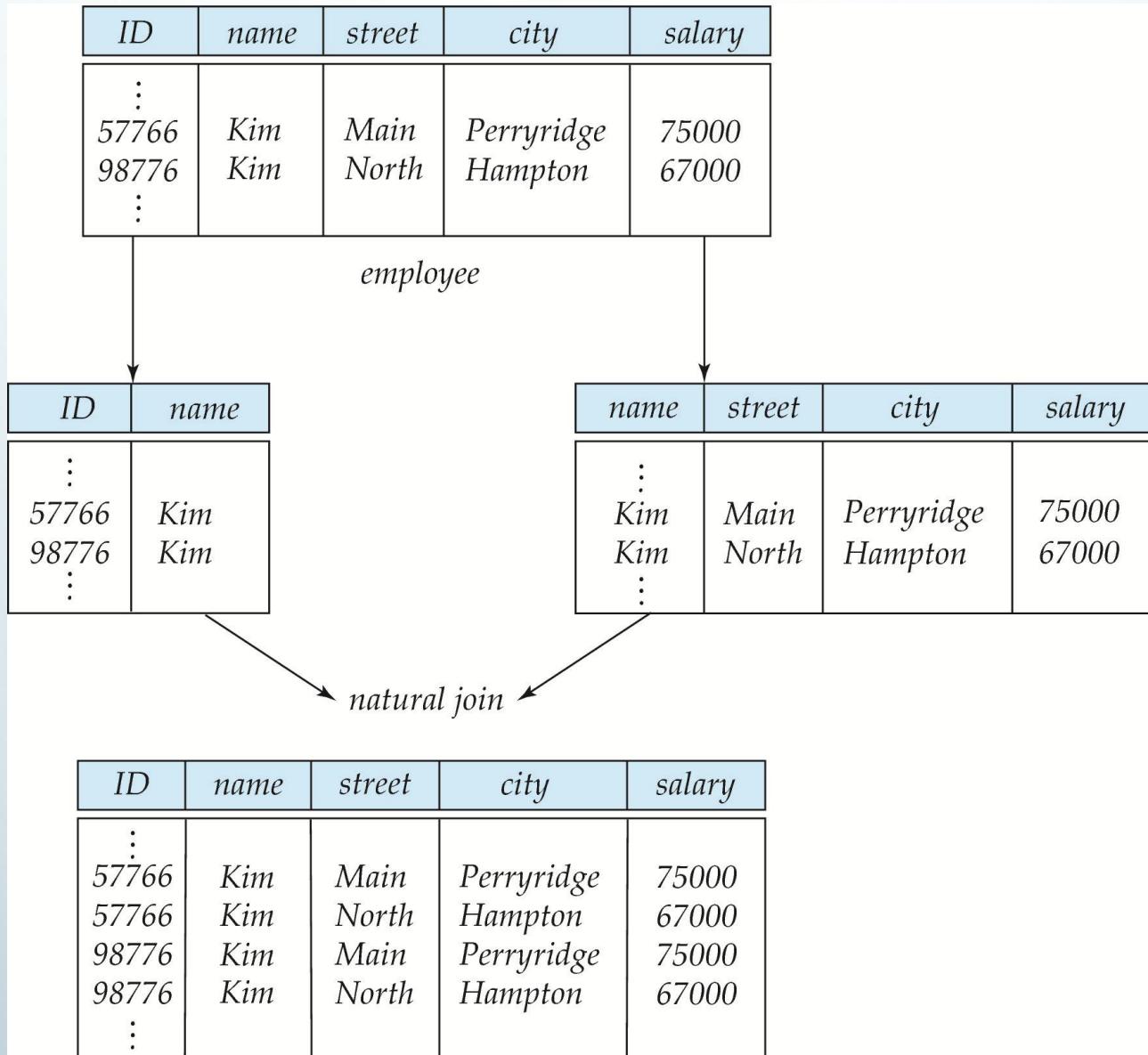
*employee2* (*name*, *street*, *city*, *salary*)

مشکل زمانی به وجود می‌آید که دو کارمند نام یکسان داشته باشند.

اسلاید بعدی نشان می‌دهد که در این حالت اطلاعات از دست می‌رود — یعنی دیگر نمی‌توانیم جدول اصلی *employee* را دوباره بازسازی کنیم — و به همین دلیل این کار یک «تجزیه دارای اتلاف» (lossy decomposition) است.



# یک تجزیه دارای اتلاف





## تجزیه بدون اتلاف

فرض کنید  $R$  یک شمای رابطه باشد و  $R_1$  و  $R_2$  یک تجزیه ( تقسیم ) از  $R$  را تشکیل دهند؛ یعنی:

$$R = R_1 \cup R_2$$

می‌گوییم این تجزیه بدون اتلاف اطلاعات **lossless** است اگر با جایگزین کردن  $R$  با دو رابطه  $R_1$  و  $R_2$  هیچ اطلاعاتی از دست نرود.

می‌گوییم این تقسیم‌بندی بدون اتلاف است اگر:

بعد از اینکه جدول  $R$  را به  $R_1$  و  $R_2$  تقسیم کردیم،

و سپس دوباره با عمل **join** این دو جدول را به هم وصل کردیم،  
 دقیقاً همان جدول اولیه  $R$  به دست بیاید.

به صورت رسمی:

$$\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r$$

و بر عکس، یک تجزیه دارای اتلاف **lossy** است اگر:

$$r \subset \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r$$



## مثال یک تجزیه بدون اتلاف

مثال  $R=(A,B,C)$  به تجزیه  $A,B$  و  $B,C$  دارد ■

$A$	$B$	$C$
$\alpha$	1	A
$\beta$	2	B

$r$

$A$	$B$
$\alpha$	1

$\Pi_{A,B}(r)$

$B$	$C$
1	A

$\Pi_{B,C}(r)$

$\Pi_A(r) \bowtie \Pi_B(r)$

$A$	$B$	$C$
$\alpha$	1	A
$\beta$	2	B



## تئوری نرمال‌سازی

- تشخیص این‌که یک رابطه (Relation) به نام  $R$  در «فرم خوب» قرار دارد یا نه.
- اگر رابطه  $R$  در فرم خوب نباشد، آن را به مجموعه‌ای از رابطه‌ها،  $R_1, R_2, \dots, R_n$  تجزیه می‌کنیم، به‌گونه‌ای که:
  - هر یک از رابطه‌های حاصل در فرم خوب باشند
  - این تجزیه بدون اتلاف اطلاعات Lossless باشد
- تئوری ما بر پایه موارد زیر است:
  - وابستگی‌های تابعی
  - وابستگی‌های چندمقداری



## وابستگی‌های تابعی

■ در دنیا واقعی معمولاً مجموعه‌ای از محدودیت‌ها (قوانين) روی داده‌ها وجود دارد.

برای مثال، برخی از محدودیت‌هایی که انتظار می‌رود در یک پایگاه داده دانشگاه برقرار باشند، عبارت‌اند از:

■ دانشجوها و استادی با شناسه ID خود به‌طور یکتاوی شناسایی می‌شوند.

یعنی هیچ دو دانشجو یا استاد نمی‌توانند ID یکسان داشته باشند.

■ هر دانشجو و هر استاد فقط یک نام دارد.

■ هر استاد و هر دانشجو (به‌طور اصلی) فقط به یک دپارتمان وابسته است.

■ هر دپارتمان فقط یک مقدار برای بودجه خود دارد و تنها یک ساختمان به آن اختصاص داده شده است.



## وابستگی‌های تابعی - ادامه

- یک نمونه **instance** از یک رابطه که تمام این محدودیت‌های دنیای واقعی را رعایت کند، «نمونه معتبر» آن رابطه نامیده می‌شود. یک پایگاه داده زمانی معتبر است که همه رابطه‌های داخل آن نمونه‌های معتبر باشند.
- این محدودیت‌ها روی مجموعه رابطه‌های معتبر اعمال می‌شوند و تعیین می‌کنند که: مقدار یک سری از ویژگی‌ها **attributes** باید مقدار مجموعه دیگری از ویژگی‌ها را به‌طور یکتا مشخص کند.
- در واقع، وابستگی تابعی یک شکل تعمیم‌یافته از مفهوم کلید **Key** است.



## مفهوم وابستگی‌های تابعی

- فرض کنید  $R$  یک اسکیما (طرح) رابطه‌ای باشد.  
 $\beta \subseteq R$  و  $\alpha \subseteq R$

- وابستگی تابعی  
 $\alpha \rightarrow \beta$

- روی  $R$  برقرار است اگر و تنها اگر برای هر نمونه قانونی ( $r(R)$ ، هرگاه دو تاپل  $t_1$  و  $t_2$  از  $r$  روی صفت‌های مجموعه  $\alpha$  با هم برابر باشند، آن دو تاپل روی صفت‌های مجموعه  $\beta$  هم با هم برابر باشند. به عبارتی:

A	B
1	4
1	5
3	7

$$t_1[\beta] = t_2[\beta] \text{ آنگاه } t_1[\alpha] = t_2[\alpha] \text{ اگر}$$

مثال: رابطه‌ای به نام  $r(A, B)$  در نظر بگیرید با یک نمونه مشخص. روی این نمونه، وابستگی  $A \rightarrow B$  برقرار است؛ اما  $B \rightarrow A$  برقرار نیست.



## بسته مجموعه‌ای از وابستگی‌های تابعی

- با داشتن یک مجموعه  $F$  از وابستگی‌های تابعی، برخی وابستگی‌های تابعی دیگر وجود دارند که به صورت منطقی از  $F$  نتیجه می‌شوند.

If  $A \rightarrow B$  and  $B \rightarrow C$ , then we can infer that ■

$$A \rightarrow C$$

- etc.

■ مجموعه تمام وابستگی‌های تابعی که از نظر منطقی از  $F$  نتیجه می‌شوند، بستار  $F$  نام دارد.

■ بستار  $F$  را با نماد  $F^+$  نمایش می‌دهیم.



## کلیدها و وابستگی‌های تابعی

کلید برای شمای رابطه  $R$  است اگر و تنها اگر بتواند همه ویژگی‌های  $R$  را تعیین کند.

کلید کاندید است برای  $R$  است اگر و تنها اگر:

- $K \rightarrow R$ , and
- for no  $\alpha \subset K$ ,  $\alpha \rightarrow R$

وابستگی‌های تابعی به ما اجازه می‌دهند قیود و محدودیت‌هایی را بیان کنیم که با استفاده از آبرکلیدها قابل بیان نیستند. برای مثال، در شمای زیر:

*in\_dep (ID, name, salary, dept\_name, building, budget).*

انتظار داریم این وابستگی‌های تابعی برقرار باشند:

$dept\_name \rightarrow building$

$ID \rightarrow building$

اما انتظار نمی‌رود که وابستگی زیر برقرار باشد:

$dept\_name \rightarrow salary$



# Use of Functional Dependencies

- ما از وابستگی‌های تابعی برای این کارها استفاده می‌کنیم:
- برای بررسی رابطه‌ها تا ببینیم آیا تحت مجموعه‌ای از وابستگی‌های تابعی معتبر هستند یا نه.
- اگر رابطه  $\alpha$  تحت مجموعه  $F$  معتبر باشد، می‌گوییم  $\alpha$ ، وابستگی‌های تابعی  $F$  را ارضاء می‌کند.
- برای مشخص کردن محدودیت‌ها روی مجموعه رابطه‌های معتبر.
- می‌گوییم  $F$  روی  $R$  برقرار است اگر تمام رابطه‌های معتبر روی  $R$ ، وابستگی‌های تابعی  $F$  را ارضاء کنند.
- نکته: ممکن است یک نمونه مشخص از یک طرح رابطه، یک وابستگی تابعی را برقرار کند، حتی اگر آن وابستگی تابعی در تمام نمونه‌های معتبر برقرار نباشد.
- برای مثال، یک نمونه خاص از رابطه  $instructor$  ممکن است به صورت اتفاقی وابستگی  $name \rightarrow ID$ .



# Trivial Functional Dependencies

- یک وابستگی تابعی زمانی بدیهی است که در تمام نمونه‌های ممکن یک رابطه برقرار باشد.
- مثال:

  - $ID, name \rightarrow ID$
  - $name \rightarrow name$

- به طور کلی  $\beta \subseteq \alpha \rightarrow \beta$  زمانی بدیهی است که



# Lossless Decomposition

- قاعده دقیق تجزیه بدون اتلاف
- برای تجزیه‌ی یک رابطه  $R$  به دو رابطه  $R_1$  و  $R_2$ , اگر حداقل یکی از شرایط زیر برقرار باشد، تجزیه بدون اتلاف است:

$$(R_1 \cap R_2) \rightarrow R_1 \text{ یا } (R_1 \cap R_2) \rightarrow R_2$$

- یعنی:
  - اشتراک دو رابطه، کلید یکی از آن‌ها باشد.

## تفسیر ساده

- اگر اشتراک بتواند کل  $R_1$  را بسازد  $\rightarrow$  بدون اتلاف
- اگر اشتراک بتواند کل  $R_2$  را بسازد  $\rightarrow$  بدون اتلاف
- نیازی نیست هر دو با هم برقرار باشند



# Example

رابطه زیر را داریم:

$$R(A, B, C)$$

وابستگی‌های تابعی:

$$F = \{ A \rightarrow B, B \rightarrow C \}$$

یعنی:

با دانستن  $A$  مقدار  $B$  مشخص می‌شود.

با دانستن  $B$  مقدار  $C$  مشخص می‌شود.

در نتیجه به صورت زنجیره‌ای:  
 $A \rightarrow C$  نیز برقرار است.



# Lossless Decomposition

مثال رابطه زیر را داریم:  $R(A, B, C)$  وابستگی‌های تابعی: ■  
حالت اول:

- $R_1(A, B), R_2(B, C)$

حالت دوم: تجزیه به  $R1$  و  $R2$

$R1(A, B)$

$R2(A, C)$



# Example

رابطه زیر را داریم:  
وابستگی‌های تابعی:

$$F = \{ A \rightarrow B, B \rightarrow C \}$$

حالت اول: تجزیه به  $R_2$  و  $R_1$

$$\begin{array}{l} R1(A, B) \\ R2(B, C) \end{array}$$

شرط تجزیه بدون اتلاف  
برای بدون اتلاف بودن تجزیه، باید اشتراک دو رابطه بتواند  
یکی از آن‌ها را تعیین کند.  
اشتراک:

$$R1 \cap R2 = \{B\}$$

بررسی وابستگی:

$$B \rightarrow BC$$

یعنی با دانستن  $B$  می‌توان کل  $R2(B, C)$  را به دست آورد.

نتیجه

این تجزیه بدون اتلاف است، چون اشتراک  $B$  یک کلید برای  $R2$  است.

حالت دوم: تجزیه به  $R_2$  و  $R_1$

$$\begin{array}{l} R1(A, B) \\ R2(A, C) \end{array}$$

شرط تجزیه بدون اتلاف  
اشتراک:

$$R1 \cap R2 = \{A\}$$

بررسی وابستگی:

$$A \rightarrow AB$$

یعنی با دانستن  $A$  می‌توان کل  $R1(A, B)$  را به دست آورد.  
نتیجه

این تجزیه هم بدون اتلاف است، چون اشتراک  $A$  یک کلید برای  $R1$  است.



# Dependency Preservation

- بررسی محدودیت‌های وابستگی تابعی هر بار که پایگاه داده به روزرسانی می‌شود می‌تواند بسیار پرهزینه باشد.
- مفید است که پایگاه داده به گونه‌ای طراحی شود که بتوان محدودیت‌ها را به صورت کارآمد و سریع بررسی کرد.
- اگر بررسی یک وابستگی تابعی تنها با استفاده از یک رابطه قابل انجام باشد، هزینه بررسی آن بسیار کم خواهد بود.
- هنگام تجزیه یک رابطه ممکن است وضعیت به گونه‌ای شود که برای بررسی یک وابستگی تابعی مجبور به انجام ضرب دکارتی باشیم.
- تجزیه‌ای که اجرای وابستگی تابعی را از نظر محاسباتی دشوار کند، «حفظ‌کننده وابستگی» محسوب نمی‌شود.



# Dependency Preservation Example

■ در یک شِما:

$dept\_advisor(s\_ID, i\_ID, dept\_name)$

■ و با وابستگی‌های تابعی زیر:

$i\_ID \rightarrow dept\_name$

$s\_ID, dept\_name \rightarrow i\_ID$

در طراحی بالا مجبور هستیم نام دپارتمان را برای هر بار که یک استاد در رابطه  $dept\_advisor$  ظاهر می‌شود، تکرار کنیم.

برای رفع این مشکل باید رابطه  $dept\_advisor$  را تجزیه کنیم.  
اما هر تجزیه‌ای انجام دهیم، تمام صفات موجود در وابستگی

$s\_ID, dept\_name \rightarrow i\_ID$

در یک رابطه واحد باقی نخواهند ماند.

به همین دلیل، ترکیب حاصل وابستگی‌ها را حفظ نمی‌کند (dependency preserving نیست).



# نرمال فرم‌ها



# Boyce-Codd Normal Form

- تعریف ساده BCNF:
- است اگر در هر وابستگی تابعی  $X \rightarrow Y$ , مجموعه یک رابطه در BCNF است اگر در هر وابستگی تابعی  $X \rightarrow Y$ , مجموعه ابرکلید باشد.



# Boyce-Codd Normal Form

- یک شمای رابطه‌ای  $R$  نسبت به مجموعه‌ای از وابستگی‌های تابعی  $F$  در نرمال فرم بویس-کاد BCNF قرار دارد اگر برای تمام وابستگی‌های تابعی موجود در  $F^+$  که به صورت:

$$\alpha \rightarrow \beta$$

هستند و  $\alpha$  و  $\beta$  هر دو زیرمجموعه‌ای از  $R$  هستند، حداقل یکی از شروط زیر برقرار باشد:

- وابستگی  $\alpha \rightarrow \beta$  بدیهی باشد (یعنی  $\beta$  زیرمجموعه‌ای از  $\alpha$  باشد)
- یک ابرکلید برای  $R$  باشد. اگر وابستگی تابعی وجود دارد باید آن ابرکلید باشد.



# Boyce-Codd Normal Form (Cont.)

*in\_dep (ID, name, salary, dept\_name, building, budget)*

- مثالِ یک شِمایی که در BCNF نیست:  
زیرا:

$dept\_name \rightarrow building, budget$

▪ در رابطه **in\_dep** برقرار است.

▪ ولی

▪ یک آبرکلید **dept\_name**

وقتی **in\_dep** را به دو رابطه **department** و **instructor** تجزیه می‌کنیم،

▪ رابطه **instructor** در BCNF است.

▪ رابطه **department** نیز در BCNF است.



# Decomposing a Schema into BCNF

- فرض میکنیم  $R$  یک شمای رابطه باشد که در BCNF نیست.
- اگر  $\alpha \rightarrow \beta$  وابستگی تابعی‌ای باشد که باعث نقض BCNF شده است.
- ما  $R$  را به دو رابطه تجزیه می‌کنیم:
  - $(\alpha \cup \beta)$  در یک رابطه جدید
  - $(R - (\beta - \alpha))$  در یک رابطه جدید
- در مثال ما یعنی `:in_dep`
  - $\alpha = dept\_name$
  - $\beta = building, budget$
- و `:in_dep` جایگزین می‌شود با:
  - $(\alpha \cup \beta) = (dept\_name, building, budget)$
  - $(R - (\beta - \alpha)) = (ID, name, dept\_name, salary)$



# Example

- $R = (A, B, C)$   
 $F = \{A \rightarrow B, B \rightarrow C\}$
- $R_1 = (A, B), R_2 = (B, C)$

- تجزیه بدون اتلاف پیوست:

$$R_1 \cap R_2 = \{B\} \text{ and } B \rightarrow BC$$

- حفظ کننده وابستگی ها:

- $R_1 = (A, B), R_2 = (A, C)$
- تجزیه بدون اتلاف پیوست:

$$R_1 \cap R_2 = \{A\} \text{ and } A \rightarrow AB$$

- غیر حفظ کننده وابستگی  
 $B \rightarrow C$  بدون محاسبه  $R_1 \bowtie R_2$  نمیتوانیم بگیم)



# Example: Is it BCNF?

$$R = (A, B, C)$$
$$F = \{A \rightarrow B, B \rightarrow C\}$$

بررسی وابستگی‌ها:

$A \rightarrow B$

( $A \rightarrow BC$  کلید است (چون  $\checkmark A$

$B \rightarrow C$

کلید نیست  $\times B$

: پس

$R(A, B, C) \notin BCNF$

چه کنیم؟



# Example

- رابطه و وابستگی‌ها
- رابطه اصلی:
- $R(A, B, C)$
- وابستگی‌های تابعی:
- $F = \{A \rightarrow B, B \rightarrow C\}$

تجزیه اول

$$R_1(A, B), R_2(B, C)$$

۱) تجزیه بدون اتصال پیوست

اشتراک:

$$R_1 \cap R_2 = \{B\}$$

• وابستگی:

$$B \rightarrow BC$$

چون **B** کلید رابطه  $(C, B, R_2)$  است، اتصال طبیعی  $R_1$  و  $R_2$  اطلاعات را از

دست نمی‌دهد.

✓ تجزیه بدون اتصال

۲) حفظ‌کننده وابستگی‌ها

- وابستگی  $A \rightarrow B$  کاملاً در  $R_1$  قابل بررسی است.
- وابستگی  $B \rightarrow C$  کاملاً در  $R_2$  قابل بررسی است.
- ✓ تمام وابستگی‌ها بدون **Join** قابل کنترل‌اند.
- ✓ تجزیه حفظ‌کننده وابستگی‌ها



# Example

- رابطه و وابستگی‌ها
- رابطه اصلی:
- $R(A, B, C)$
- وابستگی‌های تابعی:
- $F = \{A \rightarrow B, B \rightarrow C\}$

تجزیه دوم

$$\begin{aligned} R_1(A, B) \\ R_2(A, C) \end{aligned}$$

۲) غیر حفظ‌کننده وابستگی‌ها  
وابستگی  $A \rightarrow B$  در  $R_1$  موجود است.  
اما وابستگی مهم:

$$B \rightarrow C$$

در هیچ‌کدام از روابط به تنها‌ی وجود ندارد:  
در  $R_1(A, B)$  صفت **C** نیست.

در  $R_2(A, C)$  صفت **B** نیست.

بنابراین نمی‌توان وابستگی  $C \rightarrow B$  را بدون **Join** گرفتن از  $R_1$  و  $R_2$  بررسی کرد.

۱) تجزیه بدون اتصال پیوست  
اشتراک:

$$R_1 \cap R_2 = \{A\}$$

وابستگی:

$$A \rightarrow AB$$

چون **A** کلید  $R_1$  است، اتصال طبیعی بدون از دست رفتن داده انجام می‌شود.

✓ تجزیه بدون اتصال



# BCNF and Dependency Preservation

همیشه ممکن نیست که همزمان به BCNF و حفظ وابستگی‌ها برسیم.

یک شما را در نظر بگیرید:

$dept\_advisor(s\_ID, i\_ID, department\_name)$

با وابستگی‌های تابعی:

$i\_ID \rightarrow dept\_name$

$s\_ID, dept\_name \rightarrow i\_ID$

جدول  $dept\_advisor$  در BCNF نیست.

یک ابرکلید نیست.

هر تجزیه‌ای از  $dept\_advisor$  شامل همه صفات موجود در وابستگی

$s\_ID, dept\_name \rightarrow i\_ID$

نخواهد بود.

بنابراین این تجمعی وابستگی‌ها را حفظ نمی‌کند



## Third Normal Form

یک شِمای رابطه  $R$  در نرمال فرم سوم  $3NF$  است اگر برای تمام وابستگی‌های تابعی:

$$\alpha \rightarrow \beta \text{ in } F^+$$

حداقل یکی از شروط زیر برقرار باشد:

- $\beta \rightarrow \alpha$  بدبیهی است (i.e.,  $\beta \in \alpha$ )
- یک آبرکلید برای  $R$  باشد.

هر ویژگی  $A$  در  $\alpha - \beta$  دار یکی از کلیدهای کاندیدای  $R$  وجود داشته باشد.

توجه: هر ویژگی ممکن است در کلید کاندیدای متفاوتی قرار داشته باشد.

اگر یک رابطه در  $BCNF$  باشد، به طور خودکار در  $3NF$  هم هست (چون در  $BCNF$  یکی از دو شرط اول همیشه برقرار است).

شرط سوم یک نرم‌سازی حداقلی نسبت به  $BCNF$  است تا حفظ وابستگی‌ها تضمین شود (بعداً دلیلش را خواهیم دید).



## 3NF Example

یک شِما را در نظر بگیرید:

$dept\_advisor(s\_ID, i\_ID, dept\_name)$

با وابستگی‌های تابعی:

$i\_ID \rightarrow dept\_name$

$s\_ID, dept\_name \rightarrow i\_ID$

Two candidate keys =  $\{s\_ID, dept\_name\}$ ,  $\{s\_ID, i\_ID\}$

قبلًاً دیدیم که  $dept\_advisor$  در BCNF نیست.

اما این رابطه در 3NF قرار دارد.

ترکیب  $s\_ID$  و  $dept\_name$  یک آبرکلید است.

یک آبرکلید نیست اما:  $i\_ID \rightarrow dept\_name$

$\{ dept\_name \} - \{ i\_ID \} = \{ dept\_name \}$  and

در یک کلید کاندید قرار دارد. و  $dept\_name$



# Redundancy in 3NF

■ به شمای  $R$  در زیر توجه کنید که در حالت نرمال سوم 3NF قرار دارد.

- $R = (J, K, L)$
- $F = \{JK \rightarrow L, L \rightarrow K\}$
- و یک جدول نمونه:

$J$	$L$	$K$
$j_1$	$l_1$	$k_1$
$j_2$	$l_1$	$k_1$
$j_3$	$l_1$	$k_1$
null	$l_2$	$k_2$

■ مشکل این جدول چیست؟

- تکرار اطلاعات
- نیاز به استفاده از مقدارهای NULL (برای مثال، برای نمایش رابطه 2و2k زمانی که مقدار متضاد وجود ندارد)



# Comparison of BCNF and 3NF

- مزیت‌های نرمال‌فرم سوم 3NF نسبت به BCNF همیشه می‌توان یک طراحی 3NF بـه دست آورد بدون اینکه خاصیت بـی‌اتلاف بودن یا حفظ وابستگی‌ها از بین برود.

## معایب 3NF

- ممکن است مجبور شویم از مقدارهای تهی (NULL) برای نمایش برخی از روابط معنادار میان داده‌ها استفاده کنیم.
- مشکل تکرار اطلاعات وجود دارد.



# Goals of Normalization

- یک رابطه  $R$  با مجموعه‌ای از وابستگی‌های تابعی  $F$  در نظر بگیرید.
- تصمیم بگیر که آیا طرح رابطه  $R$  در «وضعیت خوب» قرار دارد یا نه.
- اگر یک طرح رابطه  $R$  در «وضعیت خوب» نباشد، باید آن را به مجموعه‌ای از طرح‌های رابطه‌ای تجزیه کنیم به‌طوری که:
  - هر طرح رابطه‌ای در وضعیت خوب باشد.
  - تجزیه، یک تجزیه بدون از دست‌رفتن اطلاعات (lossless) باشد.
  - ترجیحاً تجزیه باید وابستگی‌ها را نیز حفظ کند.



# How good is BCNF?

برخی از طرح‌های پایگاهداده که در BCNF قرار دارند، ممکن است به نظر نرسد که به طور کافی نرمال شده‌اند.

رابطه زیر را در نظر بگیرید:

**inst\_info (ID, child\_name, phone)**

در این رابطه ممکن است یک استاد بیش از یک شماره تلفن داشته باشد و همچنین چندین فرزند داشته باشد.

نمونه‌ای از داده‌ها (برای instance)

<i>ID</i>	<i>child_name</i>	<i>phone</i>
99999	David	512-555-1234
99999	David	512-555-4321
99999	William	512-555-1234
99999	William	512-555-4321



## How good is BCNF? (Cont.)

- هیچ وابستگی تابعی غیربدهی‌ای وجود ندارد و بنابراین این رابطه در BCNF قرار دارد.
- ناهنجاری‌های درج — یعنی، اگر شماره تلفن ۹۸۱-۹۹۲-۳۴۴۳ را به شناسه ۹۹۹۹۹ اضافه کنیم، مجبوریم دو رکورد اضافه کنیم:
  - (99999, David, 981-992-3443)
  - (99999, William, 981-992-3443)



# Higher Normal Forms

■ بهتر است رابطه  $\circ$  **inst\_info** را به شکل زیر تجزیه کنیم:

$inst\_child$ : •

<i>ID</i>	<i>child_name</i>
99999	David
99999	William

$inst\_phone$ : •

<i>ID</i>	<i>phone</i>
99999	512-555-1234
99999	512-555-4321

■ این موضوع نشان می‌دهد که به نرمال‌فرم‌های بالاتری مثل نرمال‌فرم چهارم  $4NF$  نیاز داریم، که در ادامه با آن آشنا خواهیم شد.



## معایب نرمال سازی

- نرمال سازی تکنیک مهمی برای طراحی پایگاه داده های کارآمد است اما در ضمنی که افزونگی داده را کاهش می دهد زیرا:
  - سبب کاهش سرعت اجرای سیستم می شود.
  - درجات بالای نرمال معمولاً جداول بیشتر را می طلبند.
  - برای پاسخ به پرس و جوها گاهی باید کلیه جداول تقسیم شده دوباره با هم الحق شوند
- در کاربردهایی که زمان پاسخ مهم است (نظیر وب) مطلوب نیست.



## معایب نرمال سازی

- بالاترین سطح نرمال سازی باید با توجه به عملیات کاربردی در نظر گرفته شود:
- در پایگاه داده هایی که بیشتر خواندنی هستند و افزونگی داده در آنها مشکل حادی نیست، مانند داده های کاتالوگ یک سایت تجارت الکترونیکی، می توان سطح نرمالسازی را کاهش داد. به این عمل **denormalization** می گویند.
- در کاربردهایی که درگیر داده های مهم مانند داده های مالی هستند که دائما در حال تغییرند و باید سازگار باقی بمانند، احتمالا سعی می شود به سطوح بالاتر نرمال برسند حتی اگر سرعت پایگاه داده کم شود.



## معایب نرمال سازی

- گاهی با توجه به وضعیت ممکن است داده ها از چند پایگاه داده نرمال شده استخراج شوند و در یک انبار داده غیر نرمال قرار گیرد.
- این روش برای مخزن داده **Data warehouse** استاندارد خوبی است.



# غیرنرم‌السازی

سناریو: برنامه تجارت الکترونیک

فرض کنید یک پایگاه داده برای یک پلتفرم تجارت الکترونیک طراحی می‌کنید با ساختار عادی شده زیر:

ProductID	ProductName	CategoryID
101	Laptop	1
102	Smartphone	2

CategoryID	CategoryName
1	Electronics
2	Mobile Phones

OrderID	Customer ID	ProductID	OrderDate
1	201	101	2024-12-01

سناریو: برنامه تجارت الکترونیک

1. جدول محصولات:

2. جدول دسته‌بندی‌ها:

3. جدول سفارش‌ها:

کوئری در ساختار عادی شده:

برای تولید گزارشی از سفارش‌ها به همراه نام محصولات و دسته‌بندی آن‌ها، به چندین اتصال نیاز دارید:

```
SELECT o.OrderID, o.CustomerID, p.ProductName,  
c.CategoryName, o.OrderDate  
FROM Orders o  
JOIN Products p ON o.ProductID = p.ProductID  
JOIN Categories c ON p.CategoryID = c.CategoryID;
```



# غیرنرم‌مال سازی

- برای بهینه‌سازی عملکرد، می‌توانید داده‌های دسته‌بندی را مستقیماً در جدول محصولات ذخیره کنید و جزئیات محصول و دسته‌بندی را به جدول سفارش‌ها اضافه کنید:

ProductID	ProductName	CategoryName
101	Laptop	Electronics
102	Smartphone	Mobile Phones

OrderID	CustomerID	ProductNa me	CategoryNa me	OrderDate
1	201	Laptop	Electronics	2024-12-01

- جدول محصولات غیرنرم‌مال شده:  
جدول سفارش‌ها غیرنرم‌مال شده:  
مزایای غیرعادی‌سازی:  
کوئری‌های ساده‌تر:
- SELECT OrderID, CustomerID, ProductName, CategoryName, OrderDate FROM Orders;
- این کوئری هیچ پیوندی ندارد.
- بیهوده‌گردانی: از آنجا که اتصالات حذف شده‌اند، بازیابی داده‌ها سریع‌تر انجام می‌شود، به‌ویژه برای مجموعه داده‌های بزرگ در محیط‌های پرترافیک.  
معایب:
  - افزونگی: اگر نام دسته‌بندی تغییر کند مثلاً "Electronics" به "Electronic Devices" ، باید در مکان‌های مختلف بهروزرسانی شود.
  - احتمال ناسازگاری داده‌ها: اگر به درستی مدیریت نشود، غیرعادی‌سازی ممکن است منجر به داده‌های ناسازگار شود.



# Functional-Dependency Theory Roadmap

- حالا به نظریه رسمی می‌پردازیم که مشخص می‌کند کدام وابستگی‌های تابعی به صورت منطقی از یک مجموعه وابستگی تابعی نتیجه می‌شوند.
- سپس الگوریتم‌هایی را ارائه می‌کنیم که تجزیه‌های بی‌اتلاف به  $3NF$  و  $BCNF$  تولید می‌کنند.
- سپس الگوریتم‌هایی را ارائه می‌کنیم تا بررسی کنیم آیا یک تجزیه، وابستگی‌ها را حفظ می‌کند یا نه.



# Closure of a Set of Functional Dependencies

- اگر مجموعه‌ای  $F$  از وابستگی‌های تابعی داشته باشیم، یکسری وابستگی‌های تابعی دیگر هم وجود دارند که به صورت منطقی از  $F$  نتیجه می‌شوند.

$A \rightarrow C$  and  $B \rightarrow C$ , ...

- مجموعه تمام وابستگی‌های تابعی که به صورت منطقی از  $F$  به دست می‌آیند، بستار  $F$  نامیده می‌شود.

بستار  $F$  را با نماد  $F^+$  نمایش می‌دهیم.



# Closure of a Set of Functional Dependencies

- می‌توانیم  $F^+$  (بستار مجموعه  $F$ ) را با اعمال مکرر اصول آرمستانگ محاسبه کنیم.
  - قاعده بازتابی:  $\text{if } \beta \subseteq \alpha, \text{ then } \alpha \rightarrow \beta$
  - قاعده تقویت (گسترش):  $\text{if } \alpha \rightarrow \beta, \text{ then } \gamma \alpha \rightarrow \gamma \beta$
  - قاعده تعدی:  $\text{if } \alpha \rightarrow \beta, \text{ and } \beta \rightarrow \gamma, \text{ then } \alpha \rightarrow \gamma$
- این قواعد «درست» هستند
  - یعنی فقط وابستگی‌هایی را تولید می‌کنند که واقعاً برقرارند.
  - و «کامل» هستند — یعنی همه وابستگی‌های تابعی‌ای را که برقرار می‌شوند تولید می‌کنند.



## Example of $F^+$

- $R = (A, B, C, G, H, I)$

$$F = \{ A \rightarrow B$$

$$A \rightarrow C$$

$$CG \rightarrow H$$

$$CG \rightarrow I$$

$$B \rightarrow H \}$$

- برخی اعضای  $F^+$  عبارت اند از:

- $A \rightarrow H$

- از این ها نتیجه میگیریم  $A \rightarrow B$  and  $B \rightarrow H$

- $AG \rightarrow I$

- نتیجه میگیریم  $AG \rightarrow CG$  با افزودن  $A \rightarrow C$  و  $G$  به  $AG$  سپس با  $CG \rightarrow I$

- $CG \rightarrow HI$

- نتیجه میگیریم  $CG \rightarrow CI$ , با افزودن  $CG \rightarrow I$

- نتیجه میگیریم  $CGI \rightarrow HI$ , و افزودن  $CG \rightarrow H$



# Closure of Functional Dependencies (Cont.)

- قوانین اضافی:
  - قانون اتحاد: اگر  $\alpha \rightarrow \beta\gamma$  برقرار باشد، آنگاه  $\alpha \rightarrow \beta$  و  $\alpha \rightarrow \gamma$  برقرار است.
  - قانون تجزیه: اگر  $\alpha \rightarrow \beta\gamma$  برقرار باشد، آنگاه  $\alpha \rightarrow \beta$  و  $\alpha \rightarrow \gamma$  برقرار است.
  - قانون شبه تراگذری: اگر  $\alpha\gamma \rightarrow \beta$  برقرار باشد، آنگاه  $\gamma\beta \rightarrow \delta$  برقرار است.
- قوانین بالا را می توان از روی اصول آرمستانگ استنتاج کرد.



# Procedure for Computing $F^+$

- برای محاسبه بسته‌ی مجموعه‌ای از وابستگی‌های تابعی  $F^+$ :

$$F^+ = F$$

تکرار کن:

برای هر وابستگی تابعی  $f$  در  $F^+$ ,

قوانین بازتابی و بسط (افزایشی) را روی  $f$  اعمال کن،  
وابستگی‌های تابعی بهدستآمده را به  $F^+$  اضافه کن.

برای هر جفت وابستگی تابعی  $f_1$  و  $f_2$  در  $F^+$ ,

اگر  $f_1$  و  $f_2$  با استفاده از قانون گذارایی قابل ترکیب باشند،  
پس وابستگی تابعی بهدستآمده را به  $F^+$  اضافه کن.

تا زمانی که  $F^+$  دیگر تغییری نکند.

- توجه: بعداً یک روش جایگزین برای این کار خواهیم دید.



# Closure of Attribute Sets

با داشتن یک مجموعه ویژگی  $a$ , بسته  $a^+$  تحت  $F$  (که با نشان داده می‌شود) مجموعه‌ای از ویژگی‌ها است که به صورت تابعی توسط  $F$  تعیین می‌شوند.

الگوریتم محاسبه  $a^+$ , یعنی بسته  $a$  تحت  $F$  :

```
result := α;  
while (changes to result) do  
for each  $β \rightarrow γ$  in  $F$  do  
    begin  
        if  $β \subseteq result$  then  $result := result \cup γ$   
    end
```



# Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$
- $(AG)^+$ 
  1.  $result = AG$
  2.  $result = ABCG$       ( $A \rightarrow C$  and  $A \rightarrow B$ )
  3.  $result = ABCGH$       ( $CG \rightarrow H$  and  $CG \subseteq AGBC$ )
  4.  $result = ABCGHI$       ( $CG \rightarrow I$  and  $CG \subseteq AGBCH$ )
- Is  $AG$  a candidate key?
  1. Is  $AG$  a super key?
    1. Does  $AG \rightarrow R?$  == Is  $R \supseteq (AG)^+$
    2. Is any subset of  $AG$  a superkey?
      1. Does  $A \rightarrow R?$  == Is  $R \supseteq (A)^+$
      2. Does  $G \rightarrow R?$  == Is  $R \supseteq (G)^+$
      3. In general: check for each subset of size  $n-1$



# Uses of Attribute Closure

• چند کاربرد برای الگوریتم بستار صفت‌ها وجود دارد:

• برای آزمون آبرکلید بودن:

برای اینکه ببینیم یک مجموعه صفت  $\alpha$  آبرکلید هست یا نه، بستار  $\alpha$  را حساب می‌کنیم ( $\alpha^+$ ), و بررسی می‌کنیم که آیا  $\alpha^+$  شامل تمام صفات  $R$  هست یا نه.

• برای آزمون وابستگی تابعی:

برای بررسی اینکه یک وابستگی تابعی  $\beta \rightarrow \alpha$  برقرار است) یا به عبارتی در  $F^+$  قرار دارد(، کافی است ببینیم  $\beta$  زیرمجموعه‌ای از  $\alpha^+$  باشد.

یعنی  $\alpha^+$  را با استفاده از الگوریتم بستار صفت‌ها محاسبه می‌کنیم، و بعد بررسی می‌کنیم که آیا شامل  $\beta$  هست یا نه.

این یک تست ساده، سریع و بسیار کاربردی است.

• برای محاسبه بستار:  $F$ :

برای هر مجموعه صفت  $\gamma$  که زیرمجموعه‌ای از  $R$  است،  $\gamma^+$  را حساب می‌کنیم، و برای هر  $S$  که زیرمجموعه‌ای از  $\gamma^+$  باشد، وابستگی تابعی  $S \rightarrow \gamma$  را به عنوان نتیجه تولید می‌کنیم.



# Canonical Cover

فرض کنید مجموعه‌ای از وابستگی‌های تابعی  $F$  روی یک شمای رابطه داشته باشیم.  
هر زمان که کاربری یک بهروزرسانی روی رابطه انجام می‌دهد، سیستم پایگاه داده باید  
مطمئن شود که این بهروزرسانی هیچ وابستگی تابعی را نقض نمی‌کند؛ یعنی تمام  
وابستگی‌های تابعی در  $F$  در وضعیت جدید پایگاه داده برقرار باشند.

اگر یک بهروزرسانی هر یک از وابستگی‌های تابعی موجود در  $F$  را نقض کند، سیستم باید آن  
بهروزرسانی را برگرداند (rollback کند).

می‌توانیم تلاش لازم برای بررسی نقض‌ها را با آزمودن یک مجموعه ساده‌شده از وابستگی‌های  
تابعی که همان بستار مجموعه اولیه را دارد کاوش دهیم.

این مجموعه ساده‌شده «پوشش کانُنیکال» (canonical cover) برای تعریف پوشش کانُنیکال، ابتدا باید ویژگی‌های اضافی (extraneous attributes) را تعریف کنیم.

یک ویژگی در یک وابستگی تابعی موجود در  $F$  اضافی محسوب می‌شود اگر بتوانیم آن را  
حذف کنیم بدون اینکه  $F^+$  تغییر کند.



## Extraneous Attributes

- حذف یک صفت از سمت چپ یک وابستگی تابعی می‌تواند آن را به یک محدودیت قوی‌تر تبدیل کند.
- برای مثال، اگر وابستگی  $AB \rightarrow C$  را داشته باشیم و  $B$  را حذف کنیم، نتیجهٔ جدید  $A \rightarrow C$  ممکن است قوی‌تر باشد.
- این نتیجهٔ قوی‌تر است چون  $A \rightarrow C$  به صورت منطقی،  $AB \rightarrow C$  را نیز نتیجهٔ می‌دهد، اما  $AB \rightarrow C$  به تنها‌یی الزاماً  $A \rightarrow C$  را نتیجهٔ نمی‌دهد.
- اما بسته به این که مجموعهٔ وابستگی‌های تابعی  $F$  چه چیزهایی هستند، ممکن است بتوانیم  $B$  را بدون مشکل از  $AB \rightarrow C$  حذف کنیم.
- برای مثال، فرض کنید:  
$$F = \{ AB \rightarrow C, A \rightarrow D, D \rightarrow C \}$$
در این صورت می‌توانیم نشان دهیم که  $F$  به صورت منطقی  $A \rightarrow C$  را نتیجهٔ می‌دهد، بنابراین  $B$  در وابستگی  $AB \rightarrow C$  یک صفت زائد (Extraneous) است.



## Extraneous Attributes (Cont.)

- حذف یک صفت از سمت راست یک وابستگی تابعی می‌تواند آن را به یک محدودیت ضعیف‌تر تبدیل کند.
- برای مثال، اگر داشته باشیم  $AB \rightarrow CD$  و  $C$  را حذف کنیم، نتیجه ممکن است ضعیف‌تر باشد:  $AB \rightarrow D$ .
- ممکن است ضعیف‌تر باشد زیرا با استفاده از فقط  $AB \rightarrow D$ ، دیگر نمی‌توانیم  $C \rightarrow$  را نتیجه‌گیری کنیم.
- اما، بسته به اینکه مجموعه  $F$  وابستگی‌های تابعی چه باشد، ممکن است بتوانیم  $AB \rightarrow CD$  به طور ایمن حذف کنیم.
- برای مثال، فرض کنید که
$$F = \{ AB \rightarrow CD, A \rightarrow C \}$$
- سپس می‌توانیم نشان دهیم که حتی پس از جایگزینی  $AB \rightarrow CD$  با  $AB \rightarrow CD$  هنوز می‌توانیم  $AB \rightarrow C$  را نتیجه‌گیری کنیم و در نتیجه.



# Extraneous Attributes

- یک صفت از یک وابستگی تابعی در  $F$  غیرضروری است اگر بتوانیم آن را حذف کنیم بدون اینکه  $F^+$  تغییر کند.
- یک مجموعه از وابستگی‌های تابعی  $F$  و وابستگی تابعی  $\alpha \rightarrow \beta$  در  $F$  را در نظر بگیرید.
- حذف از سمت چپ: صفت  $A$  در  $\alpha$  غیرضروری است اگر  $A \in \alpha$
- $F$  به طور منطقی نتیجه می‌دهد که  $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$
- حذف از سمت راست: صفت  $A$  در  $\beta$  غیرضروری است اگر  $A \in \beta$
- مجموعه وابستگی‌های تابعی  $F$  به طور منطقی  $\{(\alpha \rightarrow \beta) \rightarrow (\beta - A)\}$  نتیجه دهد.
- توجه: نتیجه‌گیری در جهت مخالف در هر یک از موارد بالا بدیهی است، زیرا یک وابستگی تابعی «قوی‌تر» همیشه یک وابستگی ضعیفتر را نتیجه می‌دهد.



# Testing if an Attribute is Extraneous

فرض کنید  $R$  یک طرح رابطه‌ای باشد و  $F$  مجموعه‌ای از وابستگی‌های تابعی باشد که روی  $R$  بقرارند.

یک صفت در وابستگی تابعی  $\beta \rightarrow \alpha$  را در نظر بگیرید.

برای آزمایش اینکه آیا صفت  $\beta \in A$  در  $\alpha$  غیرضروری است: مجموعه زیر را در نظر بگیرید:

$$F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$$

بررسی کنید که آیا  $A^+$  شامل  $A$  است یا خیر؛ اگر شامل باشد،  $A$  در  $\beta$  غیرضروری است.

برای آزمایش اینکه آیا صفت  $\alpha \in A$  در  $\beta$  غیرضروری است: بگذارید  $\gamma = A - \{\alpha\}$ . بررسی کنید که آیا  $\beta \rightarrow \gamma$  را می‌توان از  $F$  نتیجه گرفت یا خیر.

$\gamma^+$  را با استفاده از وابستگی‌های موجود در  $F$  محاسبه کنید. اگر  $\gamma^+$  شامل تمام صفت‌های  $\beta$  باشد، آنگاه  $\alpha$  در  $\beta$  غیرضروری است.



## Database System Concepts, 7<sup>th</sup> Ed.

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# Canonical Cover

یک پوشش کانونی برای  $F$  مجموعه‌ای از وابستگی‌ها  $F_C$  است به‌گونه‌ای که:

- به‌طور منطقی تمام وابستگی‌های موجود در  $F_C$  را نتیجه می‌دهد، و
- به‌طور منطقی تمام وابستگی‌های موجود در  $F$  را نتیجه می‌دهد، و هیچ وابستگی تابعی در  $F_C$  شامل صفت غیرضروری نیست، و هر سمت چپ وابستگی تابعی در  $F_C$  یکتا است. یعنی، هیچ دو وابستگی در  $F_C$  وجود ندارد
- $\alpha_2 \rightarrow \beta_2$  و  $\alpha_1 \rightarrow \beta_1$  به‌گونه‌ای که  $\alpha_1 = \alpha_2$  باشد.



# Canonical Cover

- برای محاسبه یک پوشش کانونی برای  $F$ :  
تکرار کن:
  - از قاعده اجتماع استفاده کن تا هر وابستگی در  $F$  به شکل  $\alpha_1 \rightarrow \beta_1$  و  $\alpha_2 \rightarrow \beta_2$  را با  $\alpha_1 \rightarrow \beta_1$  جایگزین کنی.
  - یک وابستگی تابعی  $\beta \rightarrow \alpha$  در  $F_C$  پیدا کن که دارای یک صفت غیرضروری در  $\alpha$  یا  $\beta$  باشد.
    - \*/ توجه: آزمایش برای صفات غیرضروری با استفاده از  $F_C$ , نه  $F$  انجام می‌شود \*
    - اگر صفت غیرضروری پیدا شد، آن را از  $\beta \rightarrow \alpha$  حذف کن.
    - تا زمانی که  $F_C$  تغییر نکند، ادامه بده.
  - توجه: پس از حذف برخی از صفات غیرضروری، ممکن است قاعده اجتماع دوباره قابل اعمال شود، بنابراین باید دوباره اعمال شود.



## Database System Concepts, 7<sup>th</sup> Ed.

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# Dependency Preservation

- فرض کنید  $F_i$  مجموعه‌ای از وابستگی‌ها در  $R_i$  باشد که فقط شامل صفات موجود در  $F^+$  هستند.
- یک تجزیه وابستگی‌ها **حفظ‌کننده وابستگی** است، اگر
$$(F_1 \cup F_2 \cup \dots \cup F_n)^+ = F^+$$
با استفاده از تعریف بالا، آزمایش برای حفظ وابستگی زمان نمایی می‌برد.
- توجه داشته باشید که اگر یک تجزیه وابستگی‌ها **حفظ‌کننده وابستگی** نباشد، آنگاه بررسی به روزرسانی‌ها برای نقض وابستگی‌های تابعی ممکن است نیازمند محاسبه جوین‌ها باشد، که هزینه‌بر است.



# Dependency Preservation (Cont.)

- فرض کنید  $F$  مجموعه وابستگی‌ها روی طرح رابطه‌ای  $R$  باشد و  $R_1, R_2, \dots, R_n$  یک تجزیه از  $R$  باشد.
- محدودسازی  $F$  به  $R_i$  مجموعه‌ای  $F_i$  است از تمام وابستگی‌های تابعی در  $F^+$  که فقط شامل صفات  $R_i$  هستند.
- از آنجا که تمام وابستگی‌های تابعی در یک محدودسازی فقط شامل صفات یک طرح رابطه‌ای هستند، امکان بررسی برآورده شدن چنین وابستگی‌ای تنها با بررسی همان رابطه وجود دارد.
- توجه داشته باشید که تعریف محدودسازی از تمام وابستگی‌ها در  $F^+$  استفاده می‌کند، نه فقط وابستگی‌های موجود در  $F$ .
- مجموعه محدودسازی‌ها  $F_1, F_2, \dots, F_n$  مجموعه‌ای از وابستگی‌های تابعی است که می‌توان آن‌ها را به‌طور مؤثر بررسی کرد.



# Testing for Dependency Preservation

برای بررسی اینکه آیا یک وابستگی  $\alpha \rightarrow \beta$  در تجزیه‌ای از  $R_1, R_2, \dots, R_n$  به  $R$  حفظ شده است، از آزمون زیر استفاده می‌کنیم (بسته صفت‌ها با توجه به  $F$  محاسبه می‌شود):

$$\text{result} = \alpha \quad \square$$

تکرار کن:

برای هر  $R_i$  در تجزیه:

$$t = (\text{result} \cap R_i)^+ \cap R_i \quad \square$$

$$\text{result} = \text{result} \cup t \quad \square$$

تا زمانی که  $\text{result}$  تغییر نکند

اگر  $\text{result}$  شامل تمام صفات  $\beta$  باشد، آنگاه وابستگی تابعی  $\alpha \rightarrow \beta$  حفظ شده است.

این آزمون را روی تمام وابستگی‌های موجود در  $F$  اعمال می‌کنیم تا بررسی کنیم آیا یک تجزیه **حفظ‌کننده وابستگی** است یا خیر.

این روش زمان چند جمله‌ای می‌برد، برخلاف زمان نمایی مورد نیاز برای محاسبه  $F^+$  و  $(F_1 \cup F_2 \cup \dots \cup F_n)^+$



## Database System Concepts, 7<sup>th</sup> Ed.

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# Testing for BCNF

برای بررسی اینکه آیا یک وابستگی غیرتبیینی  $\alpha \rightarrow \beta$  باعث نقض BCNF می‌شود:

بسته صفت‌های  $\alpha$ ، یعنی  $\alpha^+$ ، را محاسبه کنید، و

بررسی کنید که آیا شامل تمام صفات  $R$  است یا خیر، یعنی آیا  $\alpha$  یک ابرکلید برای  $R$  است.

آزمون ساده‌شده: برای بررسی اینکه آیا یک طرح رابطه‌ای  $R$  در BCNF است، کافی است فقط وابستگی‌های موجود در مجموعه داده‌شده  $F$  را برای نقض BCNF بررسی کنیم، به جای اینکه تمام وابستگی‌ها در  $F^+$  را بررسی کنیم.

اگر هیچ یک از وابستگی‌های موجود در  $F$  باعث نقض BCNF نشود، آنگاه هیچ یک از وابستگی‌های موجود در  $F^+$  نیز باعث نقض BCNF نخواهد شد.

با این حال، آزمون ساده‌شده با استفاده از  $F$  تنها، هنگام بررسی یک رابطه در تجزیه  $R$  درست نیست.

مثال: فرض کنید

$$R = (A, B, C, D, E)$$

$$F = \{ A \rightarrow B, BC \rightarrow D \}$$

$R$  را به دو رابطه تجزیه می‌کنیم:

$$R_1 = (A, B)$$

$$R_2 = (A, C, D, E)$$

هیچ یک از وابستگی‌های موجود در  $R_2 = (A, C, D, E)$  فقط شامل صفات  $F$  نیستند، بنابراین ممکن است به اشتباه فکر کنیم  $R_2$  در BCNF است.

در واقع، وابستگی  $AC \rightarrow D$  در  $F^+$  نشان می‌دهد که  $R_2$  در BCNF نیست.



# Testing Decomposition for BCNF

برای بررسی اینکه آیا یک رابطه  $R_i$  در تجزیه‌ای از  $R$  در BCNF است:

- یا  $R_i$  را با توجه به محدودسازی  $F^+$  به  $R_i$  بررسی کنید (یعنی تمام وابستگی‌های تابعی در  $F^+$  که فقط شامل صفات موجود در  $R_i$  هستند)،  
یا از مجموعه اصلی وابستگی‌ها  $F$  که روی  $R$  برقرارند استفاده کنید، اما با آزمون زیر:
  - برای هر مجموعه‌ای از صفات  $\alpha \subseteq R_i$ ، بررسی کنید که  $(\alpha^+ - \alpha)$  بسته صفت‌های ( $\alpha$  یا هیچ یک از صفات  $R_i - \alpha$ ) را شامل نمی‌شود، یا تمام صفات  $R_i$  را شامل می‌شود.
  - اگر این شرط توسط یک وابستگی  $\alpha \rightarrow \beta$  در  $F^+$  نقض شود، وابستگی  $\alpha \rightarrow (\alpha^+ - \alpha) \cap R_i$  را می‌توان نشان داد که روی  $R_i$  برقرار است، و در نتیجه  $R_i$  را نقض می‌کند.
  - از این وابستگی برای تجزیه  $R_i$  استفاده می‌کنیم.



# BCNF Decomposition Algorithm

```
result := {R};  
done := false;  
compute  $F^+$ ;  
while (not done) do  
  if (there is a schema  $R_i$  in result that is not in BCNF)  
    then begin  
      let  $\alpha \rightarrow \beta$  be a nontrivial functional dependency that  
      holds on  $R_i$  such that  $\alpha \rightarrow R_i$  is not in  $F^+$ ,  
      and  $\alpha \cap \beta = \emptyset$ ;  
      result := (result -  $R_i$ )  $\cup$  ( $R_i - \beta$ )  $\cup$  ( $\alpha, \beta$ );  
    end  
  else done := true;
```

توجه: هر  $R_i$  در BCNF است و تجزیه Lossless-join می‌باشد.



# Example of BCNF Decomposition

رابطه:

**class** (course\_id, title, dept\_name, credits, sec\_id, semester, year, building, room\_number, capacity, time\_slot\_id)

وابستگی‌های تابعی:

course\_id → title, dept\_name, credits

building, room\_number → capacity

course\_id, sec\_id, semester, year → building, room\_number,  
time\_slot\_id

کلید کاندید: course\_id, sec\_id, semester, year}

**BCNF:**

وابستگی course\_id → title, dept\_name, credits برقرار است،  
اما course\_id ابرکلید نیست.

پس رابطه **class** را جایگزین می‌کنیم با:

**course** (course\_id, title, dept\_name, credits)

**class-1** (course\_id, sec\_id, semester, year, building, room\_number, capacity, time\_slot\_id)



# BCNF Decomposition (Cont.)

- رابطه **course** در BCNF است.  
چگونه این را می‌دانیم؟
- وابستگی **class-1** در  $\text{building}, \text{room\_number} \rightarrow \text{capacity}$  برقرار است،  
اما  $\{\text{building}, \text{room\_number}\}$  ابرکلید برای **class-1** نیست.
- پس **class-1** را جایگزین می‌کنیم با:
  - **classroom** ( $\text{building}, \text{room\_number}, \text{capacity}$ )
  - **section** ( $\text{course\_id}, \text{sec\_id}, \text{semester}, \text{year}, \text{building}, \text{room\_number}, \text{time\_slot\_id}$ )
- رابطه‌های BCNF در **section** و **classroom** هستند.



# Third Normal Form

- برحی موقع وجود دارد که:
- **BCNF** وابستگی‌ها را حفظ نمی‌کند، و
- بررسی مؤثر برای نقض وابستگی‌های تابعی در هنگام بهروزرسانی اهمیت دارد.
- راه حل: تعریف یک فرم نرمال ضعیفتر به نام سومین فرم نرمال (**3NF**)
- اجازه می‌دهد مقداری افزونگی وجود داشته باشد (با مشکلات ناشی از آن؛ نمونه‌ها بعداً بررسی می‌شوند)،
- اما وابستگی‌های تابعی را می‌توان روی روابط جداگانه بررسی کرد بدون نیاز به محاسبه **join**.
- همیشه یک تجزیه بدون از دست دادن اطلاعات ( **lossless-join** ) و حفظ کننده وابستگی‌ها به **3NF** وجود دارد.



## 3NF Example -- Relation *dept\_advisor*

- رابطه:  
**dept\_advisor (s\_ID, i\_ID, dept\_name)**
- وابستگی های تابعی:  
 $F = \{ s\_ID, dept\_name \rightarrow i\_ID, i\_ID \rightarrow dept\_name \}$
- دو کلید کاندید:  
 $\{s\_ID, dept\_name\}$
- $\{i\_ID, s\_ID\}$
- رابطه **R** در ۳NF است.
- تحلیل وابستگی ها:  
 $s\_ID, dept\_name \rightarrow i\_ID$
- یک ابر کلید است.  
 $i\_ID \rightarrow dept\_name$
- در یک کلید کاندید وجود دارد.  
 $dept\_name$



# Testing for 3NF

- فقط لازم است وابستگی‌های تابعی موجود در  $F$  بررسی شوند و نیازی به بررسی تمام وابستگی‌ها در  $F^+$  نیست.
- از بسته صفت‌ها استفاده کنید تا برای هر وابستگی  $\alpha \rightarrow \beta$  بررسی کنید که آیا  $\alpha$  یک ابرکلید است یا خیر.
- اگر  $\alpha$  یک ابرکلید نباشد، باید بررسی کنیم که آیا هر صفت در  $\beta$  در یکی از کلیدهای کاندید  $R$  موجود است یا خیر.
- این آزمون نسبتاً پرهزینه است، زیرا شامل پیدا کردن کلیدهای کاندید می‌شود.
- آزمون برای  $3NF$  نشان داده شده که مسئله‌ای **NP-hard** است.
- جالب اینکه، تجزیه به سومین فرم نرمال (که به زودی توضیح داده می‌شود) می‌تواند در زمان چندجمله‌ای انجام شود.



# 3NF Decomposition Algorithm

Let  $F_c$  be a canonical cover for  $F$ ;

$i := 0$ ;

**for each** functional dependency  $\alpha \rightarrow \beta$  in  $F_c$  **do**

**if** none of the schemas  $R_j$ ,  $1 \leq j \leq i$  contains  $\alpha \beta$

**then begin**

$i := i + 1$ ;

$R_i := \alpha \beta$

**end**

**if** none of the schemas  $R_j$ ,  $1 \leq j \leq i$  contains a candidate key for  $R$

**then begin**

$i := i + 1$ ;

$R_i :=$  any candidate key for  $R$ ;

**end**

/\* Optionally, remove redundant relations \*/

**repeat**

**if** any schema  $R_j$  is contained in another schema  $R_k$

**then** /\* delete  $R_j$  \*/

$R_j = R;;$

$i = i - 1$ ;

**return**  $(R_1, R_2, \dots, R_i)$



# 3NF Decomposition Algorithm (Cont.)

الگوریتم بالا تضمین می‌کند که:

- هر طرح رابطه‌ای  $R_i$  در ۳NF است.
- تجزیه حفظ‌کننده وابستگی‌ها و بدون از دست دادن اطلاعات (**lossless-join**) است.
- اثبات صحت این روش در انتهای این ارائه آمده است (برای مشاهده، اینجا کلیک کنید).



# 3NF Decomposition: An Example

طرح رابطه‌ای:

**cust\_banker\_branch** = (customer\_id, employee\_id, branch\_name, type)

وابستگی‌های تابعی برای این طرح رابطه‌ای عبارتند از:

customer\_id, employee\_id → branch\_name, type

employee\_id → branch\_name

customer\_id, branch\_name → employee\_id

ابتدا یک پوشش کانونی (canonical cover) محاسبه می‌کنیم:

در سمت راست وابستگی اول غیرضروری است.

هیچ صفت دیگری غیرضروری نیست، بنابراین پوشش کانونی  $F_C$  به صورت زیر است:

customer\_id, employee\_id → type

employee\_id → branch\_name

customer\_id, branch\_name → employee\_id



# 3NF Decomposition Example (Cont.)

- حلقه **for** طرح‌های ۳NF زیر را تولید می‌کند:
  - $(customer\_id, employee\_id, type)$
  - $(employee\_id, branch\_name)$
  - $(customer\_id, branch\_name, employee\_id)$
- توجه کنید که  $(customer\_id, employee\_id, type)$  یک کلید کاندید از طرح رابطه‌ای اصلی است، بنابراین نیازی به اضافه کردن طرح رابطه‌ای دیگری نیست.
- در پایان حلقه **for**، طرح‌هایی مانند  $(employee\_id, branch\_name)$  که زیرمجموعه‌ای از دیگر طرح‌ها هستند شناسایی و حذف می‌شوند.
- نتیجه وابسته به ترتیب بررسی وابستگی‌های تابعی نخواهد بود.
- طرح ۳NF‌هایی و ساده‌شده عبارت است از:
  - $(customer\_id, employee\_id, type)$
  - $(customer\_id, branch\_name, employee\_id)$



# Comparison of BCNF and 3NF

- همیشه امکان دارد یک رابطه را به مجموعه‌ای از روابط تجزیه کنیم که در ۳NF باشند، به‌گونه‌ای که:
- تجزیه بدون از دست دادن اطلاعات (lossless) باشد،
  - وابستگی‌ها حفظ شوند.
- همیشه امکان دارد یک رابطه را به مجموعه‌ای از روابط تجزیه کنیم که در BCNF باشند، به‌گونه‌ای که:
- تجزیه بدون از دست دادن اطلاعات (lossless) باشد،
  - ممکن است نتوان وابستگی‌ها را حفظ کرد.



# Design Goals

- هدف از طراحی پایگاه داده رابطه‌ای عبارت است از:
  - BCNF
  - **Lossless join** از دست دادن اطلاعات (Join without loss of information)
  - **Dependency preservation** (حفظ وابستگی‌ها)
- اگر نتوانیم به این هدف‌ها برسیم، یکی از موارد زیر را می‌پذیریم:
  - عدم حفظ وابستگی‌ها
  - افزونگی به دلیل استفاده از  $NF^3$
- جالب اینکه، SQL راه مستقیمی برای مشخص کردن وابستگی‌های تابعی جز ابرکلیدها ارائه نمی‌دهد.
- می‌توان وابستگی‌های تابعی را با استفاده از **assertion** مشخص کرد، اما بررسی آن‌ها هزینه‌بر است و در حال حاضر توسط هیچ یک از پایگاه‌های داده پرکاربرد پشتیبانی نمی‌شود!
- حتی اگر یک تجزیه حفظ‌کننده وابستگی داشته باشیم، با استفاده از SQL نمی‌توانیم به‌طور مؤثر یک وابستگی تابعی را که سمت چپ آن یک کلید نیست، بررسی کنیم.



# Multivalued Dependencies (MVDs)

فرض کنید نام فرزندان و شماره تلفن اساتید را ثبت می‌کنیم:

inst\_child(ID, child\_name)

inst\_phone(ID, phone\_number)

اگر این دو طرح رابطه‌ای را ترکیب کنیم تا داشته باشیم:

inst\_info(ID, child\_name, phone\_number)

نمونه داده‌ها:

(99999, David, 512-555-1234), ۹۹۹۹۹

(99999, David, 512-555-4321)

(99999, William, 512-555-1234)

(99999, William, 512-555-4321)

این رابطه در **BCNF** است.

چرا؟



# Multivalued Dependencies

فرض کنید  $R$  یک طرح رابطه‌ای باشد و  $\beta \subseteq R$  و  $\alpha \subseteq R$  و  
(Multivalued Dependency) وابستگی چندمقداری

$\alpha \twoheadrightarrow \beta$   
روی  $R$  برقرار است اگر در هر رابطه قانونی  $r(R)$ ، برای تمام جفت‌های تاپل  $t_1$  و  $t_2$  در  $r$  به گونه‌ای که تاپل‌های  $t_1[\alpha] = t_2[\alpha]$  و  $t_3[\alpha] = t_4[\alpha]$  در  $r$  وجود داشته باشند به طوری که:

$$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$$

$$t_3[\beta] = t_1[\beta]$$

$$t_3[R - \beta] = t_2[R - \beta]$$

$$t_4[\beta] = t_2[\beta]$$

$$t_4[R - \beta] = t_1[R - \beta]$$



# MVD -- Tabular representation

▪  $\alpha \twoheadrightarrow \beta$  از (Tabular representation) نمایش جدولی

	$\alpha$	$\beta$	$R - \alpha - \beta$
$t_1$	$a_1 \dots a_i$	$a_{i+1} \dots a_j$	$a_{j+1} \dots a_n$
$t_2$	$a_1 \dots a_i$	$b_{i+1} \dots b_j$	$b_{j+1} \dots b_n$
$t_3$	$a_1 \dots a_i$	$a_{i+1} \dots a_j$	$b_{j+1} \dots b_n$
$t_4$	$a_1 \dots a_i$	$b_{i+1} \dots b_j$	$a_{j+1} \dots a_n$



## Database System Concepts, 7<sup>th</sup> Ed.

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# Example

در مثال ما:

$ID \rightarrow child\_name$

$ID \rightarrow phone\_number$

تعريف رسمي بالا قصد دارد مفهوم زیر را رسمي کند:

با داشتن یک مقدار مشخص از (ID)  $Y$ ، مجموعه‌ای از مقادیر ( $child\_name$ ) و  $Z$  مجموعه‌ای از مقادیر ( $phone\_number$ ) با آن مرتبط هستند، و این دو مجموعه تا حدی از یکدیگر مستقل هستند.

توجه:

اگر  $Y \rightarrow Z$ ، آنگاه  $Y \rightarrow Z$  نیز برقرار است.

در واقع، در نماد بالا داریم  $Z_1 = Z_2$  و نتیجه به دنبال آن می‌آید.



# Use of Multivalued Dependencies

ما از وابستگی‌های چندمقداری (**Multivalued Dependencies**) به دو روش استفاده می‌کنیم:

- برای بررسی روابط به منظور تعیین اینکه آیا آن‌ها تحت مجموعه‌ای از وابستگی‌های تابعی و چندمقداری قانونی هستند یا خیر.

- برای مشخص کردن محدودیت‌ها روی مجموعه روابط قانونی. ما فقط با روابطی سروکار خواهیم داشت که مجموعه‌ای از وابستگی‌های تابعی و چندمقداری داده شده را رعایت کنند.

- اگر یک رابطه  $r$  نتواند یک وابستگی چندمقداری داده شده را برآورده کند، می‌توانیم یک رابطه  $r'$  بسازیم که آن وابستگی چندمقداری را رعایت کند، با اضافه کردن تپل‌ها به  $r$ .



# Theory of MVDs

- از تعریف وابستگی چندمقداری می‌توان قانون زیر را نتیجه گرفت:  
اگر  $\alpha \rightarrow \beta$ , آنگاه  $\alpha \gg \beta$   
یعنی هر وابستگی تابعی نیز یک وابستگی چندمقداری است.
- بسته  $D^+$  از  $D$  مجموعه تمام وابستگی‌های تابعی و چندمقداری است که به‌طور منطقی از  $D$  نتیجه می‌شوند.
- می‌توانیم  $D^+$  را از  $D$  محاسبه کنیم، با استفاده از تعاریف رسمی وابستگی‌های تابعی و وابستگی‌های چندمقداری.
- این نوع استدلال برای وابستگی‌های چندمقداری بسیار ساده کافی است، که به نظر می‌رسد در عمل رایج‌ترین هستند.
- برای وابستگی‌های پیچیده‌تر، بهتر است با استفاده از **سیستم قواعد استنتاج** درباره مجموعه‌ای از وابستگی‌ها استدلال کنیم.  
**(Appendix C)**



# Fourth Normal Form

یک طرح رابطه‌ای  $R$  در **4NF** نسبت به مجموعه  $D$  از وابستگی‌های تابعی و چندمقداری است، اگر برای تمام وابستگی‌های چندمقداری در  $D^+$  به شکل  $\alpha \rightarrow \beta$  و  $\beta \subseteq R$  هستند، حداقل یکی از موارد زیر برقرار باشد:

$(\alpha \cup \beta = R \text{ یا } \beta \subseteq \alpha)$  تبیینی است (یعنی **trivial**)

یک ابرکلید برای طرح رابطه‌ای  $R$  است

اگر یک رابطه در **4NF** باشد، آنگاه در **BCNF** نیز هست.



# Restriction of Multivalued Dependencies

- محدودسازی  $D_i$  به مجموعه  $R_i$  است که شامل موارد زیر می‌باشد:
  - تمام وابستگی‌های تابعی در  $D^+$  که فقط شامل صفات موجود در  $R_i$  هستند
  - تمام وابستگی‌های چندمقداری به شکل
$$\alpha \twoheadrightarrow (\beta \cap R_i)$$
  - که  $\alpha \twoheadrightarrow \beta$  و  $\alpha \subseteq R_i$  قرار دارد.



# 4NF Decomposition Algorithm

*result* := { $R$ };

*done* := false;

*compute*  $D^+$ ;

Let  $D_i$  denote the restriction of  $D^+$  to  $R_i$

**while** (*not done*)

**if** (there is a schema  $R_i$  in *result* that is not in 4NF) **then**

**begin**

    let  $\alpha \rightarrow\!\!\!\rightarrow \beta$  be a nontrivial multivalued dependency that holds

        on  $R_i$  such that  $\alpha \rightarrow R_i$  is not in  $D_i$ , and  $\alpha \cap \beta = \emptyset$ ;

*result* := (*result* -  $R_i$ )  $\cup$  ( $R_i$  -  $\beta$ )  $\cup$  ( $\alpha, \beta$ );

**end**

**else** *done* := true;

( lossless-join) هر  $R_i$  در 4NF است و تجزیه بدون از دست دادن اطلاعات می باشد.





## Database System Concepts, 7<sup>th</sup> Ed.

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# Further Normal Forms

- وابستگی‌های **Join**، وابستگی‌های چندمقداری را تعمیم می‌دهند و به فرم نرمال پروژه-جوین (PJNF) منجر می‌شوند (که همچنین به آن پنجمین فرم نرمال گفته می‌شود).
- یک کلاس از محدودیت‌های حتی عمومی‌تر وجود دارد که به یک فرم نرمال به نام **Domain-Key Normal Form** منجر می‌شود.
- مشکل این محدودیت‌های تعمیم‌یافته این است که:
  - استدلال درباره آن‌ها سخت است،
  - هیچ مجموعه قوانین استنتاج صوتی و کامل وجود ندارد.
  - به همین دلیل، این محدودیت‌ها به ندرت استفاده می‌شوند.



# Overall Database Design Process

ما فرض کرده‌ایم که طرح رابطه‌ای  $R$  از قبل داده شده است.

- ممکن است هنگام تبدیل یک نمودار E-R به مجموعه‌ای از جدول‌ها ایجاد شده باشد.
- ممکن است یک رابطهٔ واحد باشد که شامل تمام صفت‌هایی است که برای ما اهمیت دارند (که به آن رابطهٔ *universal relation* یا *جهانی* گفته می‌شود).
- فرایند نرمال‌سازی،  $R$  را به رابطه‌های کوچک‌تر تجزیه می‌کند.
- ممکن است حاصل یک طراحی اتفاقی و بدون روش مشخص باشد که سپس آن را بررسی کرده و به فرم نرمال تبدیل می‌کنیم.



# ER Model and Normalization

وقتی یک نمودار **E-R** با دقت طراحی شود و تمامی موجودیت‌ها به درستی شناسایی شده باشند، جدول‌هایی که از نمودار **E-R** تولید می‌شوند نباید نیاز به نرمال‌سازی بیشتر داشته باشند.

با این حال، در یک طراحی واقعی (و غیرکامل)، ممکن است **وابستگی‌های تابعی** از صفات غیرکلیدی یک موجودیت به سایر صفات همان موجودیت وجود داشته باشد.

مثال:

یک موجودیت کارمند که دارای صفت‌های

نام بخش

ساختمان

باشد، و **وابستگی تابعی**:

نام بخش → ساختمان

طراحی خوب، «بخش» را باید به عنوان یک موجودیت جدا در نظر می‌گرفت.

وجود **وابستگی‌های تابعی** از صفات غیرکلیدی در یک مجموعه رابطه (relationship set) نیز ممکن است، اما نادر است — زیرا اغلب رابطه‌ها **دوتاوی** (binary) هستند.



# Denormalization for Performance

ممکن است برای دستیابی به کارایی بهتر، بخواهیم از یک طرح غیرنرمال شده استفاده کنیم.  
برای مثال، نمایش پیش‌نیازها همراه با شناسه درس و عنوان درس، نیازمند انجام یک جوین بین جدول درس و جدول پیش‌نیاز است.

**راه حل ۱:** استفاده از یک رابطه *denormalized* که شامل صفت‌های درس و پیش‌نیاز باشد، همراه با تمام صفت‌های موردنیاز.  
مزایا:

جستجوی سریع تر  
معایب:

استفاده بیشتر از فضا  
زمان اضافی برای اجرای عملیات به روزرسانی  
نیاز به کدنویسی بیشتر توسط برنامه‌نویس و احتمال خطا در این کد اضافی

**راه حل ۲:** استفاده از یک نمای مادی‌سازی‌شده (**materialized view**) که به صورت ترکیب درس و پیش‌نیاز تعریف شده باشد.

مزایا و معایب مشابه مورد بالا است،  
با این تفاوت که:

نیازی به کدنویسی اضافی توسط برنامه‌نویس نیست  
احتمال خطا در کد نیز از بین می‌رود



# Other Design Issues

برخی جنبه‌های طراحی پایگاه داده با نرمال‌سازی قابل تشخیص نیستند.  
نمونه‌هایی از طراحی بد پایگاه داده که باید از آن‌ها پرهیز کرد:

بهجای استفاده از رابطه<sup>۰</sup>:

**earnings(company\_id, year, amount)**

استفاده شود از جداولی مانند:

**.earnings\_2004,earnings\_2005,earnings\_2006**

همه<sup>۰</sup> این‌ها با شمای ( **company\_id, earnings** ) تعریف شده‌اند.

این طرح‌ها اگرچه در **BCNF** قرار دارند،  
اما نجام پرس‌وجو ( **query** ) در بین سال‌های مختلف را دشوار می‌کنند  
و هر سال نیاز به ایجاد یک جدول جدید دارند.

نمونه<sup>۰</sup> دیگر:

**company\_year(company\_id, earnings\_2004, earnings\_2005, earnings\_2006)**

این جدول نیز در **BCNF** قرار دارد،  
اما مانند نمونه<sup>۰</sup> قبل:

پرس‌وجو بین سال‌ها را سخت می‌کند

هر سال نیاز به افزودن یک صفت جدید (ستون جدید) دارد

این نوع طراحی یک **crosstab** است،  
که در آن مقادیر یک صفت، تبدیل به نام ستون‌ها می‌شوند.

این نوع ساختار معمولاً در صفحه‌گستردها ( **spreadsheets** ) و ابزارهای تحلیل داده استفاده می‌شود،

اما در طراحی پایگاه داده رابطه‌ای انتخاب مناسبی نیست.



# Modeling Temporal Data

داده‌های زمانی (Temporal Data) دارای یک بازه زمانی معتبر هستند؛ یعنی دوره‌ای که داده‌ها در آن دوره صحیح و معتبر به شمار می‌آیند.

یک **snapshot**، مقدار داده‌ها در یک لحظه خاص از زمان است.

پیشنهادهای متعددی برای گسترش مدل ER با افزودن زمان اعتبار (valid time) مطرح شده‌اند؛ مانند افزودن زمان به:

**صفات (attributes)**)

مانند: آدرس یک استاد در زمان‌های مختلف

**موجودیت‌ها (entities)**)

مانند: دوره زمانی که یک دانشجو به عنوان یک موجودیت وجود دارد

**روابط (relationships)**)

مانند: بازه زمانی که یک استاد، مشاور یک دانشجو بوده است

اما هیچ استاندارد پذیرفته شده‌ای برای این گسترش وجود ندارد.



## Modeling Temporal Data (Cont.)

- ۱) در عمل، طراحان پایگاه داده معمولاً ویژگی‌های زمان شروع و پایان را به روابط اضافه می‌کنند.
- ۲) برای مثال، رابطه‌ی course(course\_id, course\_title) به شکل course(course\_id, course\_title, start, end) جایگزین می‌شود.
- ۳) محدودیت: هیچ دو تاپل نباید بازه‌های زمانی معتبر همپوشان داشته باشند.
- ۴) اجرای کارآمد این محدودیت دشوار است.
- ۵) ارجاع کلید خارجی ممکن است به نسخه فعلی داده باشد، یا به داده در یک نقطه خاص زمانی.
- ۶) برای نمونه، ریزنمرات دانشجو باید به اطلاعات درس در زمانی که درس گذرانده شده است ارجاع دهد.



## پایان فصل ۷



# Correctness of 3NF Decomposition Algorithm

- الگوریتم تجزیه به ۳NF ویژگی حفظ وابستگی‌ها را دارد، زیرا برای هر وابستگی تابعی در FC یک رابطه ساخته می‌شود.
- این تجزیه بدون اتلاف است.
- یک کلید نامزد (C) در یکی از رابطه‌های Ri حاصل از تجزیه قرار می‌گیرد.
- بستار این کلید نامزد تحت FC باید شامل تمام ویژگی‌های موجود در R باشد.
- با دنبال کردن مراحل الگوریتم بستار ویژگی می‌توان نشان داد که برای هر تاپل در Ri در نتیجه join فقط یک تاپل متناظر وجود خواهد داشت.



## Correctness of 3NF Decomposition Algorithm (Cont.)

- ادعا: اگر رابطه  $R_i$  در تجزیه‌ای باشد که توسط الگوریتم بالا ساخته شده است، آنگاه  $R_i$  در ۳NF قرار دارد.
- اثبات:

فرض کنید  $R_i$  از وابستگی  $\alpha \rightarrow \beta$  ساخته شده باشد.  
فرض کنید  $B \rightarrow \gamma$  یک وابستگی تابعی غیر بدیهی روی  $R_i$  باشد.  
( فقط لازم است وابستگی‌هایی را در نظر بگیریم که سمت راستشان یک تک‌ویژگی است).  
اکنون،  $B$  می‌تواند در  $\beta$  باشد یا در  $\alpha$ ، اما نمی‌تواند در هر دو قرار گیرد.  
هر حالت را جداگانه بررسی کنید



# Correctness of 3NF Decomposition (Cont.)

■ حالت ۱: اگر  $B$  در  $\beta$  باشد:

اگر  $\gamma$  یک سوپر کلید باشد، شرط دوم ۳NF برقرار است.

در غیر این صورت،  $a$  باید دست کم یک ویژگی داشته باشد که در  $\gamma$  وجود نداشته باشد.  
از آنجا که  $B \rightarrow^+ \gamma$  قرار دارد، باید بتوان آن را با استفاده از  $F_C$  و از طریق بستار  $\gamma$  به دست آورد.

بستار  $\gamma$  نمی‌تواند از  $a \rightarrow \beta$  استفاده کرده باشد؛ زیرا اگر از آن استفاده می‌کرد،  $a$  باید در بستار  $\gamma$  قرار می‌گرفت،

و این ممکن نیست چون فرض کردہ‌ایم  $\gamma$  یک سوپر کلید نیست.

اکنون با استفاده از  $(\beta - \{B\}) \rightarrow a$  و  $\gamma \rightarrow B$  می‌توانیم  $a \rightarrow B$  را نتیجه بگیریم (زیرا  $\gamma$  زیرمجموعه  $\alpha\beta$  است، و  $B \rightarrow \gamma$  نیست، چون  $B \rightarrow \gamma$  یک وابستگی غیر بدیهی است).

در این صورت  $B$  در سمت راست  $\beta \rightarrow \alpha$  زائد خواهد بود؛ و این ممکن نیست چون  $\alpha \rightarrow \beta$  در  $F_C$  قرار دارد.

بنابراین، اگر  $B$  در  $\beta$  باشد، آنگاه  $\gamma$  باید یک سوپر کلید باشد و شرط دوم ۳NF برقرار می‌شود.



# Correctness of 3NF Decomposition (Cont.)

اگر  $B$  در  $\alpha$  باشد: Case 2 ■

چون  $\alpha$  یک کلید کاندیدا است، گزینه سوم در تعریف ۳NF به طور واضح برقرار است.

در واقع، نمی‌توانیم نشان دهیم که یک سوپرکلید است.  
این دقیقاً دلیل وجود گزینه سوم در تعریف ۳NF را نشان می‌دهد.

Q.E.D.



# First Normal Form

دامنه (Domain) اتمی است اگر عناصر آن به عنوان واحدهای تجزیه‌ناپذیر در نظر گرفته شوند.

- نمونه‌هایی از دامنه‌های غیراتمی:

- مجموعه‌ای از نام‌ها، صفات مرکب

- شماره‌های شناسایی مانند CS101 که می‌توان آن‌ها را به بخش‌هایی تقسیم کرد

- یک طرح رابطه‌ای  $R$  در اولین فرم نرمال ۱ (NF) قرار دارد اگر دامنه تمام صفات آن اتمی باشد.

- مقادیر غیراتمی ذخیره‌سازی را پیچیده می‌کنند و باعث تشویق به ذخیره‌سازی تکراری و اضافی داده‌ها می‌شوند.

- مثال:

- مجموعه‌ای از حساب‌ها که برای هر مشتری ذخیره شده است

- مجموعه‌ای از مالکین که برای هر حساب ذخیره شده است

- ما فرض می‌کنیم که تمام روابط در اولین فرم نرمال قرار دارند (و این موضوع در فصل ۲۲: پایگاه‌های داده مبتنی بر شیء دوباره بررسی خواهد شد).



## First Normal Form (Cont.)

- اتمی بودن در واقع ویژگی نحوه استفاده از عناصر دامنه است.
- مثال: رشته‌ها معمولاً به عنوان واحدهای تجزیه‌ناپذیر در نظر گرفته می‌شوند.
- فرض کنید به دانشجویان شماره دانشجویی داده می‌شود که رشته‌هایی به شکل CS0012 یا EE1127 هستند.
- اگر دو حرف اول استخراج شوند تا دپارتمان مشخص شود، آنگاه دامنه شماره‌های دانشجویی اتمی نیست.
- انجام چنین کاری ایده خوبی نیست، زیرا باعث می‌شود اطلاعات در برنامه کاربردی کدگذاری شود، نه در پایگاه داده.