

به نام آنکه جان را فکرت آموخت



بخش ششم: معماری پایگاه داده‌ها

مرتضی امینی

نیمسال اول ۱۴۰۱-۱۴۰۰

(محتویات اسلایدها برگرفته از یادداشت‌های کلاسی استاد محمدتقی روحانی رانکوهی است.)



نیاز به یک معماری واحد از دیدگاه **داده شناسانه** (و نه دیدگاه عملکردی یا دیدگاه مولفه-مبنا) که در آن

داده‌ها به گونه‌ای قابل فهم (مستقل از پیچیدگی‌های سطح سمپاد) به کاربر نمایش داده شود.

عدم وجود اتفاق نظر در چگونگی معماری پایگاه داده‌ها در سالهای آغازین ایجاد

پیشنهاد معماری سه سطحی از سوی ANSI / SPARC

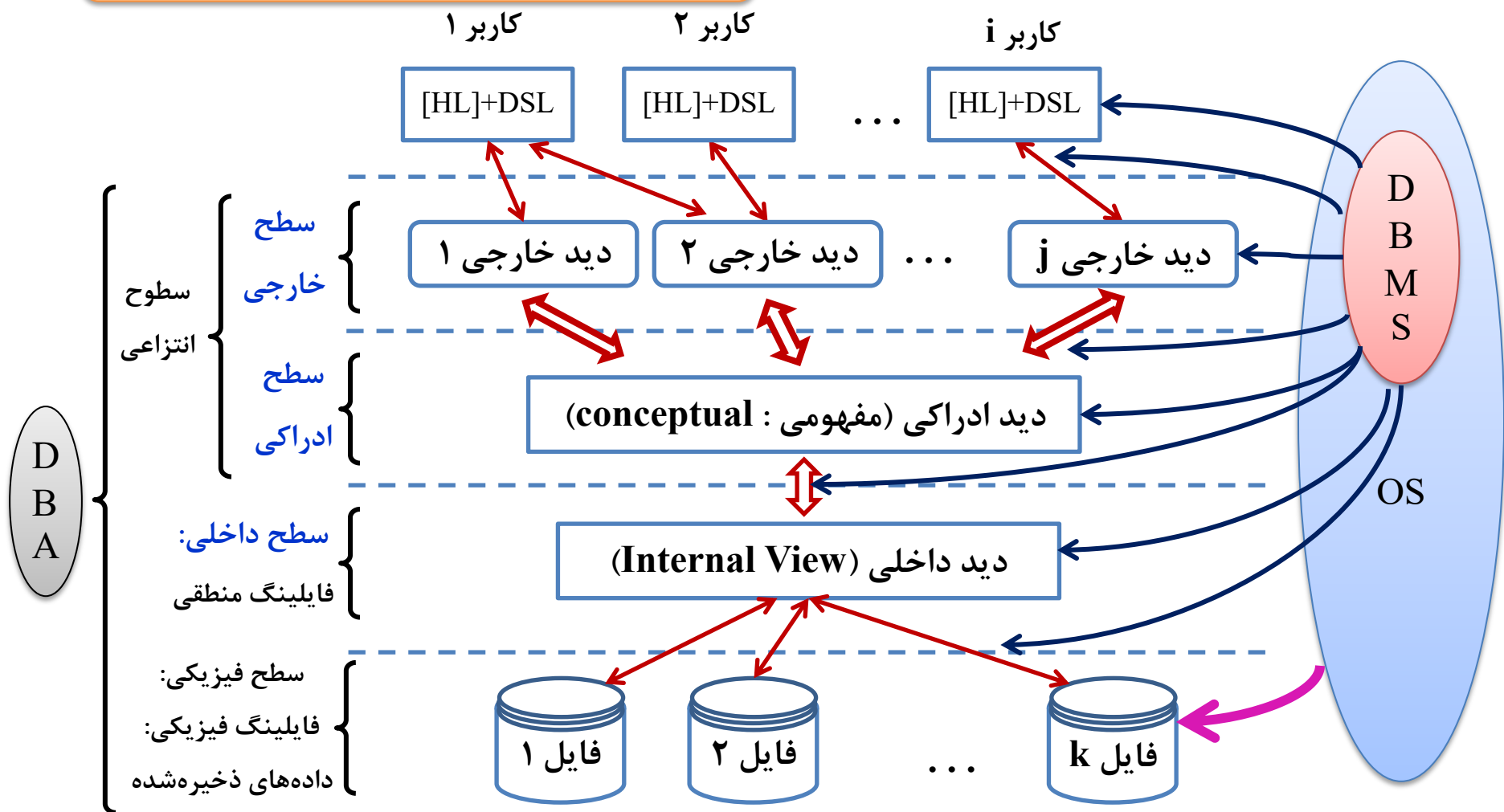
سه سطح معماری ANSI، در واقع سه سطح **تعریف و کنترل داده‌ها** است.

دو سطح در محیط انتزاعی و یک سطح در محیط فایلینگ منطقی.



نشان دهنده نگاشت (تبدیل) بین سطوح

معماری سه سطحی






□ اجزای معماری سه سطحی پایگاه داده‌ها:

- ۱- کاربر User
- ۲- زبان میزبان HL: مانند زبانهای جاوا، C# و PHP
- ۳- زبان داده‌ای فرعی (زیرزبان داده‌ای) DSL: زبانهای داده‌ای ادغام شده در زبانهای میزبان
-
- ۴- دید خارجی (نمای خارجی) → سطح خارجی
- ۵- دید ادراکی (فرایافتی یا مفهومی) → سطح ادراکی
- ۶- دید داخلی → سطح داخلی
-
- ۷- فایل‌های فیزیکی
- ۸- سیستم مدیریت پایگاه داده‌ها (کوته‌تر: سمپاد)
- ۹- مدیر پایگاه داده‌ها (DBA)



□ دید (نمای) ادراکی (فرایافتی یا مفهومی)

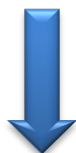
دید طراحی نسبت به داده‌های ذخیره‌شده (و نهایتاً ذخیره‌شده) در پایگاه داده‌ها 


✓ دیدی جامع: دربرگیرنده نیازهای همه کاربران محیط

✓ مطرح در محیط انتزاعی (فرافایلی) ← مبتنی بر یک ساختار داده مشخص

✓ طراحی با عنصر (عناصر) ساختاری اساسی

✓ پس از طراحی ← توصیف شود ← **شمای ادراکی (Conceptual Schema)**



 نوعی برنامه حاوی دستورات $\left. \begin{array}{l} \text{DDL} \\ \text{DCL} \end{array} \right\}$ و نه دستورات **DML**

✓ شمای ادراکی به سیستم مدیریت داده می‌شود و در کاتالوگ آن نگهداری می‌شود.



بخش ششم: معماری پایگاه داده‌ها

CREATE TABLE S1 ...

CREATE TABLE S2 ...

CREATE TABLE S3 ...



□ دید ادراکی: جدول‌های مبنای S1 و S2 و S3

□ شمای ادراکی: تعریف جدول‌ها است.

□ به این جدول‌ها **جدول‌های مبنا** می‌گوییم.

□ اطلاعات شمای ادراکی به سیستم مدیریت داده می‌شود و در کاتالوگ آن نگهداری می‌شود.

کاتالوگ سیستم

در سیستم‌های جدولی: در تعدادی
جدول که خود سیستم ایجاد می‌کند.

در کجا ؟

چگونه ؟

کاتالوگ سیستم : متا داده‌ها (Data Dictionary : Meta Data)



✓ تمامی اطلاعات شمای ادراکی

□ حاوی : ✓ داده‌های کنترلی

✓ Data About Data



بخش ششم: معماری پایگاه داده‌ها

جدول systables:



systables	نام جدول	ایجاد کننده	تاریخ ایجاد	تعداد ستون	کلید اصلی	...

کاربر پیاده‌ساز : **CREATE TABLE STT ...**

سیستم : **INSERT INTO SYSTABLES**
VALUES ('STT' , 'c1' , 'd1' , 5 , 'STID' , ...)

کاربر پیاده‌ساز : **DROP TABLE STCOT ...**

سیستم : **DELETE FROM SYSTABLES**
WHERE TNAME = 'STCOT'




افزودن ستون به یک جدول :  **ALTER TABLE STT** : کاربر پیاده‌ساز


ADD SADDRESS CHAR (80)

سیستم : **UPDATE SYSTABLES**

SET ColN = 6

WHERE TNAME = 'STT'

سیستم برای جدولی که تعداد ستون‌های آن تغییر می‌کند در سطح فایلینگ چگونه عمل می‌کند؟ 

آیا با دستور **DELETE** در سطح جدول‌های مبنا، جدول کاتالوگ تغییر می‌کند؟ 

DELETE FROM STT

WHERE STID='777'



□ دید (نمای) داخلی



دید خود DBMS [و نیز طراح پایگاه داده‌ها، در مرحله طراحی فیزیکی]، نسبت به داده‌های

ذخیره‌شده

- ✓ مطرح در سطح فایلینگ منطقی
- ✓ مبتنی بر یک [یا چند] ساختار فایل ←
 - 1:1 } پیش فرض (یک جدول : یک فایل)
 - 1:N } (چند جدول : یک فایل)
 - N:1 } (یک جدول : چند فایل)
- ✓ سطحی که فایل‌های منطقی پایگاه داده‌ها تعریف می‌شود.

✓ تناظر بین «ساخت» های سطح ادراکی و «ساخت» های سطح داخلی

Table	TableFile
STT	STTFile
COT	COTFile
STCOT	STCOTFile
...	...

تناظر 1:1 بین ساخت‌های سطح ادراکی و سطح داخلی





بخش ششم: معماری پایگاه داده‌ها



نوعی برنامه که توسط خود DBMS (و گاه براساس **اطلاعاتی** که طراح - پیاده‌ساز به سیستم می‌دهد) تولید می‌شود و شرح و وصف فایلینگ منطقی پایگاه داده‌ها است.



توجه: در شمای داخلی انواع رکوردها تعریف می‌شوند و دستورهای لازم جهت ایجاد فایل‌ها و کنترل آنها در این شما وجود دارد.

TYPE STUDENT = **RECORD**

STUDENT-ID	: String ;
STUDENT-NAME	: String ;
STUDENT-LEV	: String ;
STUDENT-MJR	: String ;
STUDENT-DEPT	: String ;

شمای داخلی ساده شده در یک زبان شبه پاسکال





□ اطلاعاتی که طراح-پیاده ساز در خصوص شاخص‌ها به سیستم می‌دهد، در دید داخلی تاثیر می‌گذارد.

□ **شاخص** نمونه‌ای است از افزونگی تکنیکی برای افزایش سرعت بازیابی داده‌ها

□ می‌توان بر روی یک (یا چند) ستون از جدول شاخص تعریف کرد.

□ وجود شاخص بر روی مقادیر یک ستون باعث می‌شود به رکوردهای حاوی شرط WHERE در فایل

مربوط به یک جدول با سرعت بیشتری دسترسی یافت.

□ **ویژگی‌های ستون شاخص:**

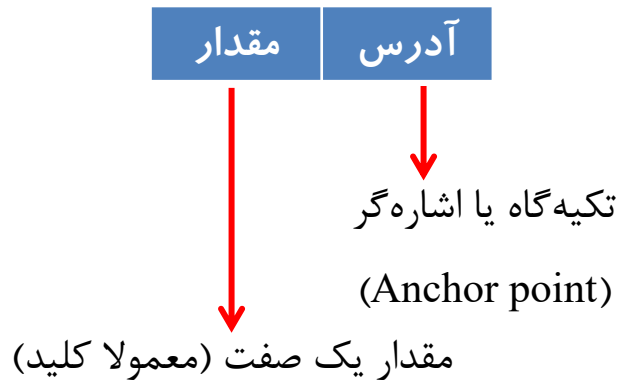
✓ تغییر ناپذیر (حتی الامکان)

✓ پرکاربرد در شرط WHERE

✓ ... ؟



□ هر شاخص تشکیل شده از تعدادی درایه (مدخل-entry)

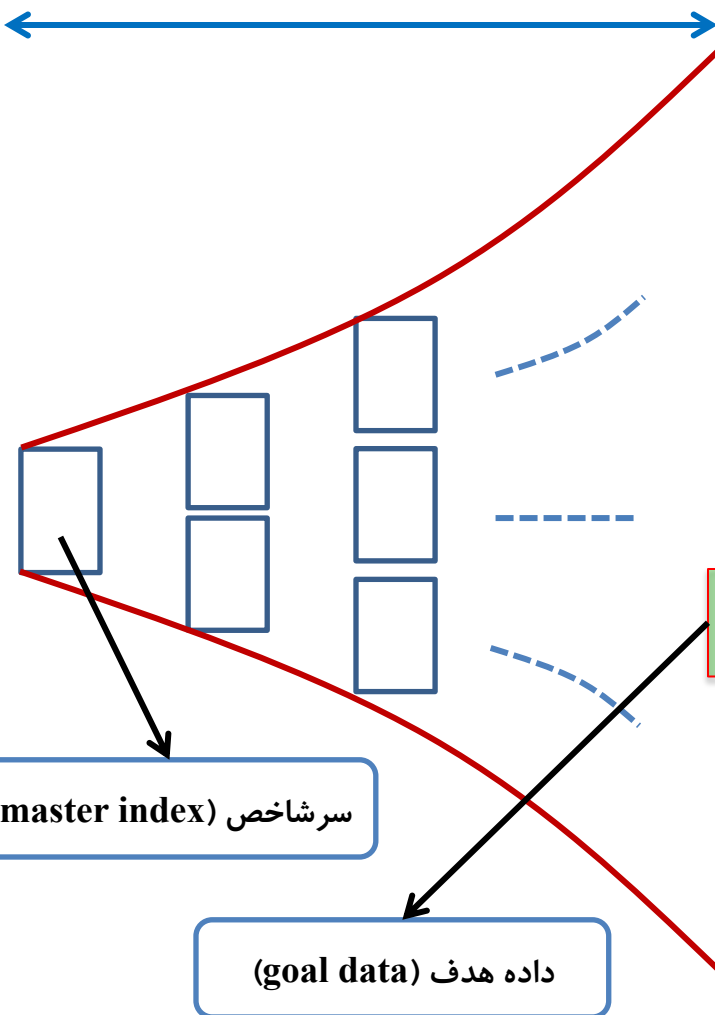


هر مدخل اشاره دارد به { یک یا گروهی (به صورت چند سطحی) } از رکوردها



فایل شاخص گذاری شده

(چندسطحی معمولاً با ساختار B-Tree یا B⁺-Tree)



...	رشته	نام	شماره
	نرم افزار		۱۰۰
	نرم افزار		۱۰۱
	سخت افزار		۱۰۲
	نرم افزار		۱۰۳
	سخت افزار		۱۰۴
	نرم افزار		۱۰۵
			⋮
	نرم افزار		<i>k</i>
			⋮
	سخت افزار		۹۹۷
	سخت افزار		۹۹۸
	نرم افزار		۹۹۹



□ انواع شاخص‌ها

□ شاخص B-Tree

✓ برای انواع شرط‌های قیاسی و بازه‌ای کاربرد دارد.

□ شاخص B⁺-Tree

✓ برای انواع شرط‌های قیاسی و بازه‌ای کاربرد دارد.

□ شاخص مبتنی بر درهم‌سازی (Hash)

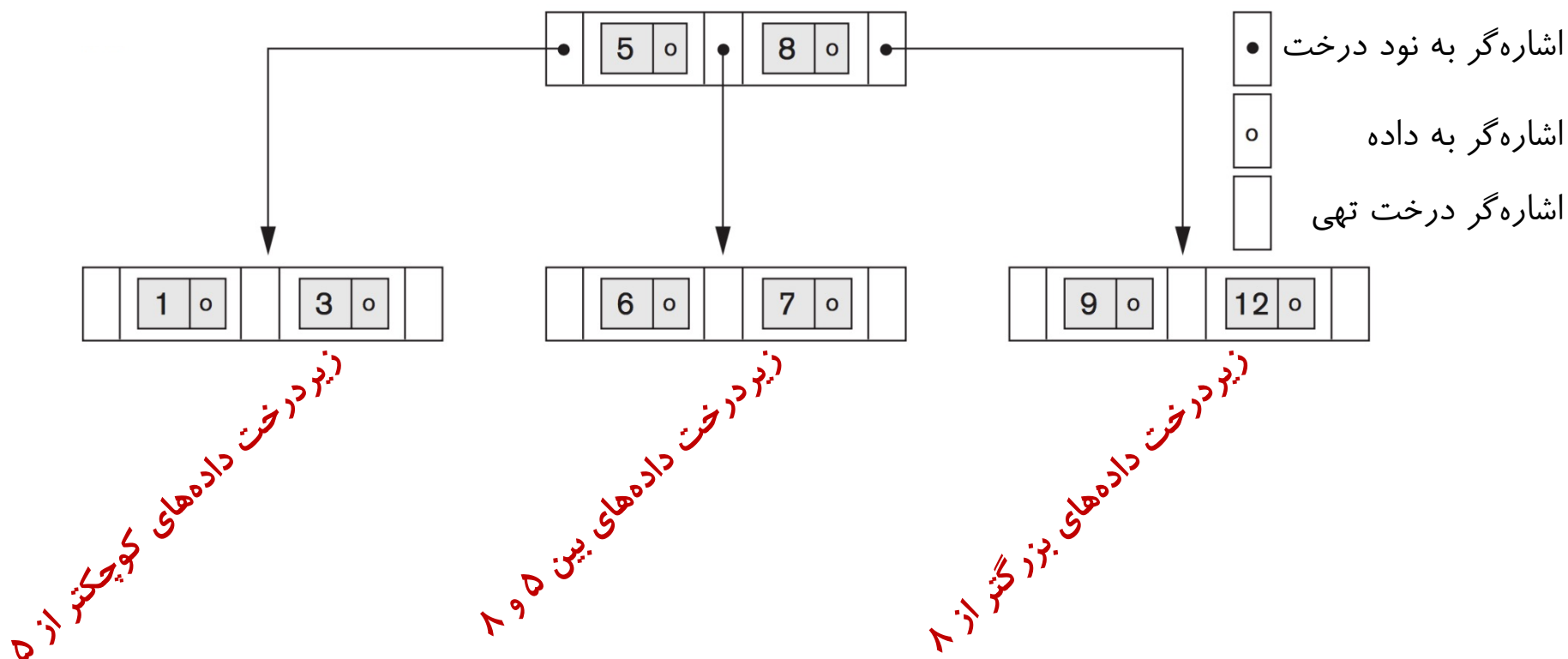
✓ وقتی کارایی دارد که فقط شرط تساوی بر روی مقادیر ستون شاخص‌گذاری شده داشته باشیم.

انواع دیگر شاخص؟





مثال شاخص B-Tree: داده‌ها به ترتیب ۸، ۵، ۱، ۷، ۳، ۱۲، ۹، ۶ درج شده‌اند.



شاخص B⁺-Tree چیست و چه مزایا و معایبی نسبت به B-Tree دارد؟





Bucket 0

13646	•
21124	•
.....	

Bucket 1

23402	•
81165	•
.....	

Bucket 2

51024	•
12676	•
.....	

Bucket 3

62104	•
71221	•
.....	

⋮

Bucket 9

34723	•
41301	•
.....	

Emp_id	Lastname	Sex
.....		
12676	Marcus	M	..
.....		
13646	Hanson	M	..
.....		
21124	Dunhill	M	..
.....		
23402	Clarke	F	..
.....		
34723	Ferragamo	F	..
.....		
41301	Zara	F	..
.....		
51024	Bass	M	..
.....		
62104	England	M	..
.....		
71221	Abercombe	F	..
.....		
81165	Gucci	F	..
.....		

شاخص مبتنی بر

درهم سازی (Hash)



تابع درهم ساز:

مجموع ارقام Emp_id در پیمانه ۱۰

با استفاده از تابع درهم ساز هر مقدار Emp_id به یک باکت نگاشت می شود.

هر باکت، مقادیر نگاشت شده به همراه اشاره گر به رکوردهای داده مربوطه را در خود دارد.



بخش ششم: معماری پایگاه داده‌ها

□ در سیستم‌های جدولی: خود سیستم روی کلید اصلی (PK) شاخص خودکار (Automatic Index) ایجاد می‌کند (عمدتاً B-Tree).

□ برای ایجاد شاخص روی دیگر ستون‌ها پیاده‌ساز باید با استفاده از دستور **CREATE INDEX** درخواست ایجاد شاخص نماید.

مثال □ ایجاد شاخص بر روی ستون STNAME که PK نیست:

CREATE INDEX SNX

ON STT (STNAME)

[CLUSTERED] ؟ خوشه‌بندی

B-Tree Index

CREATE INDEX SNX

**ON STT
USING HASH (STNAME)**

Hash Index



DROP INDEX SNX

حذف شاخص:



در سیستم چه اتفاقی می‌افتد؟

DROP TABLE
DROP INDEX

با اجرای دستور



مثالی از وضعیتی بیان کنید که براساس آن طراح-پیاده ساز تصمیم به ایجاد شاخص می‌گیرد.





دید منطقی DBMS نسبت به داده‌های ذخیره شده

۱۹

بخش ششم: معماری پایگاه داده‌ها

- ✓ چه فایل‌هایی دارد
 - ✓ نگاشت سطح ادراکی به سطح داخلی
 - ✓ صفحات (Pages) فضای پایگاه داده کاربر
 - ✓ فرمت رکورد هر فایل [رکورد داخلی]
 - ✓ ساختار هر فایل
 - ✓ کلید(ها)
 - ✓ استراتژی دستیابی به رکوردها
 - ✓ توالی منطقی رکوردها در صفحات
 - ✓ اندازه جاری هر فایل
 - ✓ اندازه گسترش فایل
 - ✓ اطلاعات همگانی
 - ✓ ارتباط منطقی بین فایل‌ها
 - ✓
- BOF
- EOF
- R/W
- ⋮

می‌داند: جنبه های فایلینگ منطقی [مجازی]

DBMS □

نمی‌داند: جنبه های فایلینگ فیزیکی



دید منطقی DBMS نسبت به داده‌های ذخیره شده (ادامه)

بخش ششم: معماری پایگاه داده‌ها

۲۰

می‌داند: جنبه های فایلینگ منطقی [مجازی] ...

- ☐ چگونگی نشست فایل‌ها روی دیسک
- ☐ استراتژی دستیابی چگونه پیاده‌سازی شده‌اند.
- ☐ اندازه بلاک (Block) فیزیکی
- ☐ جزئیات تکنیک‌های Blocking
- ☐ Locality رکوردهای فایل‌ها
- ☐ توالی منطقی رکوردها چگونه پیاده‌سازی شده‌اند
- ☐ ...

نمی‌داند: جنبه های فایلینگ فیزیکی

DBMS ☐

چيست و بر کدام یک از عملیات روی فایل‌ها تاثیر می‌گذارد؟





□ در بعضی از سیستم‌های مدیریت جدید، سیستم مدیریت، کل فضای پایگاه داده را به صورت **مجموعه‌ای** از **مجموعه صفحات** می‌بیند، یعنی نوعی **نمای مجازی** از داده‌های ذخیره شده در پایگاه داده دارد.

در سطح فایلینگ مجازی $DB = \{ Pages \}$



شماره صفحات	تعداد صفحات	نام جدول
p1 ... p10	10	STT
p15 ... p29	15	COT
P101 ... P1000	900	STCOT

SELECT STT.*

FROM STT

WHERE STID = '444'

DBMS : READ P1

(فرض کنید '444' در P1 است)



□ دید (نمای) خارجی



دید کاربر (برنامه ساز) خاص است نسبت به داده‌های ذخیره شده [مثلا دید یک AP نویس]

✓ دید جزئی (Partial): دربرگیرنده نیازهای داده‌ای یک کاربر مشخص [برای یک AP مشخص]

✓ مطرح در سطح انتزاعی ← مبتنی بر یک ساختار داده‌ای مشخص



آیا این ساختار داده همان ساختار داده سطح دید ادراکی است؟



✓ روی دید ادراکی طراحی و تعریف می‌شود.

✓ } یک کاربر ← چند دید متفاوت
چند کاربر ← یک دید مشترک



✓ توصیف دید خارجی ← **شِمای خارجی (External Schema)**



 نوعی «برنامه» که کاربر سطح خارجی می‌نویسد، حاوی دستورات «تعریف داده‌ها» و **معدود** دستورات «کنترل داده‌ها» ( چرا معدود؟)

✓ **شِمای خارجی** ← ذخیره در کاتالوگ

در سیستم‌های جدولی، دید خارجی خود نوعی جدول است، اما **مجازی (Virtual Table)** و نه ذخیره‌شده



دید خارجی در واقع **پنجره‌ای** است که از آن کاربر خارجی محدوده‌ی داده‌ای خود را می‌بیند و نه بیشتر.





بخش ششم: معماری پایگاه داده‌ها

کاربر ۱

V1

STID STNAME

777 st7

444 st4

⋮

⋮

چند ستون از یک جدول

V2

CONUM

COTITLE

دگرنامی ستون



STT

STID

STNAME

STLEV

STMJR

STDEID

777

st7

bs

phys

d11

888

st8

ms

math

d12

444

st4

bs

comp

d14

⋮

⋮

⋮

⋮

⋮

COT

COID

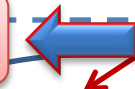
...

STCOT

STID

COID ...

تناظر یک به یک



STT FILE

COT FILE

STCOT FILE



بخش ششم: معماری پایگاه داده‌ها

کاربر ۲

v1

STID STNAME

777

st7

دید مشترک با کاربر ۱

444

st4

⋮

⋮



STT

STID	STNAME	STLEV	STMJR	STDEID
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
⋮	⋮	⋮	⋮	⋮

COT

COID	...

STCOT

STID	COID ...

STT FILE

COT FILE

STCOT FILE



بخش ششم: معماری پایگاه داده‌ها

کاربر ۳

v3

STNUM STNAME COTITLE TR YR

دید روی بیش از یک جدول



STT

STID	STNAME	STLEV	STMJR	STDEID
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
:	:	:	:	:

COT

COID	...

STCOT

STID	COID ...

STT FILE

COT FILE

STCOT FILE



بخش ششم: معماری پایگاه داده‌ها

کاربر ۴

v4

STID

STNAME

TR

YR

AVG

صفت مجازی



STT

STID

STNAME

STLEV

STMJR

STDEID

777

st7

bs

phys

d11

888

st8

ms

math

d12

444

st4

bs

comp

d14

:

:

:

:

:

COT

COID

...

STCOT

STID

COID ...

STT FILE

COT FILE

STCOT FILE



□ از این مثال‌ها نتیجه می‌گیریم که تعریف، طراحی و توصیف دید خارجی در سیستم‌های جدولی از پویایی بالایی برخوردار است.

□ یعنی انواع جدول‌های مجازی را می‌توان روی لایه‌های زیرین تعریف کرد.

□ تعریف شمای خارجی کاربر ۱ (با استفاده از مفهوم **دید**):

```
CREATE VIEW V1 [(STID, STNAME)]
AS SELECT STT.STID, STT.STNAME
FROM STT;
```

```
CREATE VIEW V2 [(SN, SJ, SL)]
AS SELECT STID, STJ, STL
FROM STT
WHERE STJ != 'phys;
[WITH CHECK OPTION]
```

شرط تعریف دید

□ در شرط تعریف دید می‌توان از نام ستونی که در محدوده دید نیست استفاده کرد (مثلاً با تعریف شرط بر روی ستون DEID از جدول STT در مثال فوق که در دید V2 نیامده است).



هر کاربر با اجازه Admin می‌تواند دید (View) خودش را داشته باشد (Sub-database).



دستور SELECT در متن دستور تعریف دید اجرایی نیست بلکه اعلانی است



یعنی هیچ داده‌ای بازیابی نمی‌شود و صرفاً برای اعلام محدوده داده‌ای کاربران است. □

تا آنجا که به تعریف دید مربوط است هر دستور SELECT معتبر با هر میزان پیچیدگی را می‌توان در



CREATE VIEW نوشت.

□ **تمرین:** مثال کاتالوگ پیش‌دیده را به نحوی گسترش دهید که اطلاعات (نه داده‌های) شمای داخلی و

شمای خارجی دیده شده را بتوان در آن ذخیره کرد (جدول دیگری برای کاتالوگ تعریف کنید که بتوان

این شماها را در آن ذخیره کرد).



نگاشت بین سطوح در عملیات سطح شمای خارجی

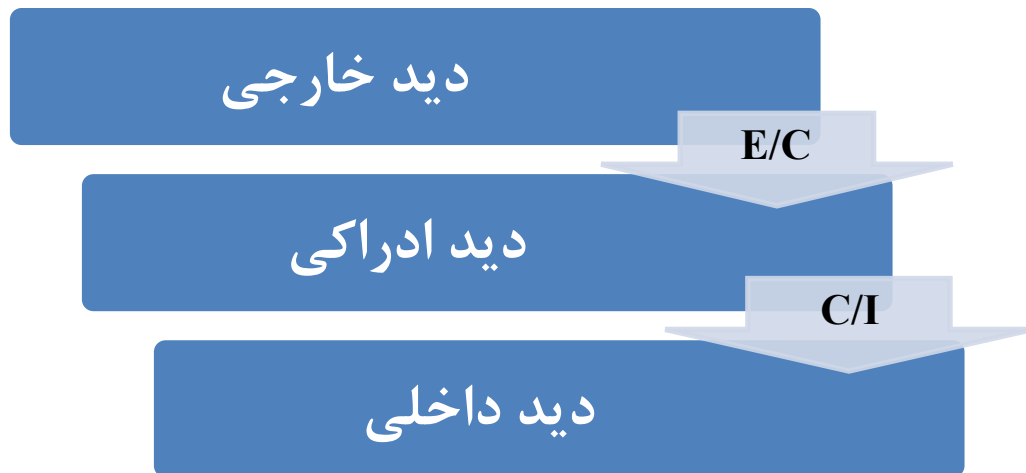
بخش ششم: معماری پایگاه داده‌ها

۳۰

جهت انجام عملیات از دید خارجی نیاز به نگاشت یا تبدیل این عملیات بین سطوح مختلف است: 

External to Conceptual Mapping :E/C 

Conceptual to Internal Mapping :C/I 



آیا تبدیل دیگری هم متصور است؟





بازیابی: کاربر حق دارد در محدوده دید خود عمل بازیابی انجام دهد.

❑ عملیات در شمای خارجی

درج

حذف

بروزرسانی

ذخیره‌سازی: به تشخیص Admin مجاز به انجام است.

❑ هر دستور [حکم] عمل‌کننده در شمای خارجی (روی دید خارجی)،

❑ تبدیل می‌شود به دستور(های) عمل‌کننده در شمای ادراکی (روی دید ادراکی)

❑ و سپس به قطعه برنامه‌ای عمل‌کننده در شمای داخلی (روی دید داخلی)

❑ و نهایتاً به عملیاتی در فایل‌های فیزیکی.



عملیات بازیابی: چون دید خارجی در سیستم‌های جدولی، به هر حال نوعی جدول است، برای بازیابی

از همان دستور SELECT استفاده می‌کنیم.

```
SELECT V2.SN  
FROM V2  
WHERE SL='ms'
```

E/C

□ سیستم در نگاشت E/C، می‌تواند شرط یا شرایط داده شده در تعریف دید را AND می‌کند با شرط یا

شرایط داده شده در پرس‌وجوی روی دید و یا در قسمت FROM، عبارت SELECT مربوط به تعریف دید را

قرار دهد. به این عمل، گاه **محاسبه دید** (View Computation) هم می‌گویند.

```
SELECT STT.STID  
FROM STT  
WHERE STL='ms'  
AND STJ != 'phys'
```

```
SELECT STID  
FROM (SELECT STID, STJ, STL  
FROM STT WHERE STJ != 'phys')  
WHERE STL='ms'
```

C/I



بخش ششم: معماری پایگاه داده‌ها

به واحد رکورد

ناحیه پیام بافر سیستم

OPEN STFILE (R, SysBuf, MessageArea, ...)

LREAD STFILE ON STLINDEX.value='ms';

...

IF SysBuf.STJ != 'phys'

MOVE SysBuf.STID INTO UBuf[SN]

...

LOOP Control;

فایلینگ منطقی
در محیط

PSEEK

جستجوی فیزیکی

PREAD

خواندن فیزیکی

فایلینگ فیزیکی
در محیط

به واحد بلاک



□ لزوماً از همه انواع دیده‌ها نمی‌توان عملیات ذخیره‌سازی در DB انجام داد.

□ همه انواع دیده‌ها قابل بروزرسانی (Updatable) نیستند.

□ محدودیتهایی هم در عمل و تاحدی در تئوری وجود دارد.

□ دید از نظر قابلیت عملیات ذخیره‌سازی (بستگی دارد به ساختار دید و مکانیزم تعریف آن):

□ پذیرا (Updatable): می‌توان از آنها عملیات ذخیره‌سازی انجام داد ولی گاه مشکلاتی دارند.

□ ناپذیرا (Non Updatable): تبدیل E/C انجام شدنی نیست.

تعریف شده روی یک جدول مبنا

□ دید

تعریف شده روی بیش از یک جدول مبنا ← در عمل ناپذیرا، اما در تئوری بعضی‌ها پذیرا هستند.



عملیات ذخیره سازی از دید تعریف شده روی یک جدول مبنا

بخش ششم: معماری پایگاه داده ها

۳۵

□ دید تعریف شده روی یک جدول مبنا

□ دید دارای کلید جدول مبنا (Key Preserving) ← پذیرا (در عمل و تئوری) اما مشکلاتی هم دارد.

□ دید فاقد کلید جدول مبنا (Non Key Preserving) ← ناپذیرا

□ دید دارای ستون [صفت] مجازی (دیدهای آماری) ← ناپذیرا



عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده‌ها

۳۶

دید حافظ کلید تعریف شده روی یک جدول مبنا ☐

V2

SN	SL	SJ
888	ms	math
444	bs	comp
⋮	⋮	

STT

STID	STNAME	STL	STJ	STD
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
⋮	⋮	⋮	⋮	⋮

```
CREATE VIEW V2 [(SN, SJ, SL)]
AS SELECT STID, STJ, STL
FROM STT
WHERE STJ != 'phys'
[WITH CHECK OPTION]
```



عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده‌ها

۳۷

□ فرض بر مجاز بودن کاربر (از نظر سیستم کنترل دسترسی) به انجام عمل داریم و لذا صرفاً شدنی بودن را بررسی می‌کنیم.

□ در دید حافظ کلید انجام عملیات سطری امکان‌پذیر است.

□ زیرا تناظر یک به یک بین سطرهاى دید و سطرهاى جدول مبنا برقرار است.

DELETE FROM V2

WHERE SN='444'

E/C

DELETE FROM STT

WHERE STID='444' AND STJ != 'phys'



حذف سطر در دید حافظ کلید

□ الان این سطر از جدول STT حذف می‌شود و اگر کاربر دیگری این سطر را در دیدش داشته باشد، دیگر به این سطر دسترسی ندارد.



عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده‌ها

۳۸

مثال
بروزرسانی در دید حافظ کلید

UPDATE V2

SET SJ='IT'

WHERE SN='444'



UPDATE STT

SET STJ='IT'

WHERE STID='444' ' AND STJ != 'phys'



عملیات ذخیره سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده ها

۳۹



از نظر تئوریک درخواست زیر به دلیل **عدم رعایت محدودیت دید** باید رد شود.

UPDATE V2

```
SET SJ='phys'  
WHERE SN='888'
```

❑ در عمل: اگر از عبارت **[with check option]** استفاده کنیم، سیستم رد می کند، وگرنه درخواست

انجام می شود اما ...

E/C

UPDATE STT

```
SET STJ='phys'  
WHERE STID='888' ' AND STJ != 'phys'
```

❑ حال اگر بنویسیم:

SELECT V2.* FROM V2

❑ سطر با کلید 888 دیگر در دید کاربر نمی آید!



عملیات ذخیره سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده ها

۴۰

درج در دید حافظ کلید



```
INSERT INTO V2  
VALUES ('555', 'chem', 'bs')
```

E/C

```
INSERT INTO STT  
VALUES ('555', ?, 'chem', 'bs', ?)
```

❑ **عدم رعایت هیچ مقدار ناپذیری ستونهای نهان از دید:** اگر هر کدام از ستونهای نهان از دید کاربر،

محدودیت هیچ مقدار ناپذیری داشته باشند، درخواست رد می شود.

❑ **عدم رعایت محدودیت یکتایی مقادیر کلید:** اگر به جای 555 بنویسیم 777، درخواست رد می شود

(تبدیل E/C انجام نمی شود) به دلیل **عدم رعایت محدودیت یکتایی** مقادیر کلید.

❑ حال اگر به جای chem بنویسیم phys، همان پیش می آید که در مثال UPDATE دیدیم.



عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده‌ها

۴۱

☐ دلایل رد شدن درخواست عمل ذخیره‌سازی در دید تک جدولی حافظ کلید:

☐ عدم رعایت محدودیت دید (در صورت وجود WITH CHECK OPTION در تعریف دید)

☐ عدم رعایت محدودیت یکتایی مقادیر کلید

☐ عدم رعایت محدودیت هیچ‌مقدارناپذیری ستون‌های نهان

☐ ...



عملیات ذخیره سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده ها

۴۲

□ دید تعریف شده روی یک جدول مبنا و فاقد کلید

چون این دید فاقد کلید است، امکان انجام عملیات سطری وجود ندارد.



CREATE VIEW V3

AS SELECT STNAME, STJ

FROM STT

□ درخواست زیر انجام نمی شود، چون معلوم نیست کدام سطر از رابطه باید حذف شود. پس تبدیل E/C

ناممکن است، مگر اینکه بپذیریم این درخواست به صورت مکانیکی انجام شود؛ یعنی تمام سطرهای

حائز شرط داده شده (مجموعه ای از سطرها) حذف شوند.

DELETE FROM V3

WHERE STNAME='ali'

□ اگر کاربر این پیامد را بپذیرد مشکلی نیست، اما در عمل سیستم ها نمی پذیرند!

□ در دید V3 انجام INSERT نیز غیرممکن است (به دلیل عدم وجود مقدار کلید برای جدول مبنا).



عملیات ذخیره سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده ها

۴۳

حال اگر در تعریف V3، DISTINCT بزنیم چه پیش می آید؟



```
CREATE VIEW V3  
AS SELECT DISTINCT STNAME, STJ  
FROM STT
```

❑ فرقی نمی کند، باز هم همان مشکل پابرجاست:

```
DELETE FROM V3  
WHERE STNAME='a'
```

E/C

تبدیل می شود به حذف مجموعه ای از سطرها



عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده‌ها

۴۴

دید تعریف شده روی یک جدول مبنا دارای ستون مجازی ☐

این دیدها هم در عمل و هم در تئوری ناپذیرا هستند. ☐



V4	PN	SQ
	P1	100
	P2	210
	P3	80

```
CREATE VIEW V4 (PN, SQ)
AS SELECT P#, SUM(QTY)
FROM SP
GROUP BY P#
```

SP	S#	P#	QTY
	S1	P1	100
	S1	P2	140
	S2	P3	80
	S2	P2	70



عملیات ذخیره‌سازی از دید تعریف شده روی یک جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده‌ها

۴۵

□ انجام عملیات سطری در دید V4 غیرممکن است.

DELETE FROM V4

WHERE PN='p1'



سطری نیست، با نوعی تفسیر
می‌توان گفت که مجموعه‌ای از
سطرها را حذف می‌کند.

□ از لحاظ تئوریک هم دید V4 نباید پذیرا باشد.

□ زیرا جدول V4 (که مجازی است) و جدول مبنای SP با هم تعارض معنایی (Semantic Conflict)

دارند. یعنی مسند بیانگر معنای رابطه V4 اساساً با مسند بیانگر رابطه SP تفاوت دارد.

[دربحث رابطه‌ای خواهیم دید که هر رابطه (جدول) یک معنا دارد و در اینجا این دو رابطه هیچ ربطی از نظر معنایی با هم ندارند].



عملیات ذخیره‌سازی از دید تعریف شده روی چند جدول مبنا

بخش ششم: معماری پایگاه داده‌ها

۴۶

دیده‌های تعریف شده روی بیش از یک جدول

در عمل این دیده‌ها **ناپذیرا** هستند و دخالت خود برنامه‌ساز لازم است. البته به لحاظ **تئوری** در

برخی موارد **پذیرا** هستند.

V5: T1 JOIN T2 (پیوند طبیعی) دید پیوندی

V6: T1 UNION T2

V7: T1 INTERSECT T2

V8: T1 EXCEPT T2

PK-PK: ستون پیوند در هر دو جدول PK است. در تئوری پذیرا و بدون مشکل

PK-FK: در تئوری پذیرا به شرط پذیرش پیامدها

FK-FK

(Non-Key) NK-NK

دید پیوندی



عملیات ذخیره سازی از دید تعریف شده روی چند جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده ها

۴۷

□ دید پیوندی PK-PK (ستون پیوند در هر دو جدول کلید اصلی است)

V5 همان STT است اما این بار به صورت یک دید تعریف شده است.



V5

STID	STNAME	STL	STJ	STD
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
⋮	⋮	⋮	⋮	⋮

CREATE VIEW V5

AS SELECT *

FROM ST1 NATURAL JOIN ST2

ST1

STID	STNAME	STL
777	st7	bs
888	st8	ms
444	st4	bs
⋮	⋮	⋮

ST2


STID	STJ	STD
777	phys	d11
888	math	d12
444	comp	d14
⋮	⋮	⋮



عملیات ذخیره‌سازی از دید تعریف شده روی چند جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده‌ها

۴۸

یک دستور اجراشونده در شمای خارجی تبدیل می‌شود به دو دستور در شمای ادراکی. 

INSERT INTO V5

VALUES ('999', 'St9', 'chem', 'bs', 'D15')

E/C

INSERT INTO ST1

VALUES ('999', 'St9', 'bs')

INSERT INTO ST2

VALUES ('999', 'chem', 'D15')

عمل DELETE در این دید تبدیل می‌شود به دو عمل حذف از جدول‌های مبنایی زیرین و عمل UPDATE 

(بسته به ستونی که می‌خواهیم بروز کنیم) به یک یا دو عمل بهنگام‌سازی در جدول‌های زیرین تبدیل

می‌شود.



عملیات ذخیره سازی از دید تعریف شده روی چند جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده ها

۴۹

□ دید پیوندی **PK-FK** (ستون پیوند، در یکی کلید اصلی و در دیگری کلید خارجی است)



CREATE VIEW V6

AS SELECT STT.STID, STT.NAME, STCOT.*

FROM STT JOIN STCOT

□ درج در این دید تبدیل می شود به درج یک تاپل ناقص در STT به شرط آنکه شماره دانشجویی تکراری نباشد. ولی در STCOT حتما یک تاپل درج می شود.

INSERT INTO V6

VALUES ('9212345', 'Amir', '40638', 15)

E/C

INSERT INTO STT

VALUES ('9212345', 'Amir', ?, ?, ?)

INSERT INTO STCOT


VALUES ('9212345', '40638', 15)




عملیات ذخیره‌سازی از دید تعریف شده روی چند جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده‌ها

۵۰

حذف از این دید مشکل دارد. 

 اگر از هر دو جدول حذف شود، منجر به حذف داده‌های ناخواسته می‌شود.

 با حذف یک سطر از جدول STT، برای حفظ جامعیت ارجاعی نیز لازم است یک تعداد سطر دیگر از

STCOT حذف شود، مگر آنکه فقط از STCOT حذف کنیم و از STT سطر مربوطه را حذف نکنیم.

 عمل به‌نگام‌سازی نیز مساله مشابه حذف ممکن است داشته باشد.

ستون پیوند در هیچ کدام کلید
نیست (نه اصلی و نه خارجی).

ستون پیوند در هر دو جدول
کلید خارجی است.

دید حاصل از پیوند FK-FK و دید حاصل از پیوند NK-NK چه رفتاری در عملیات ذخیره‌سازی دارند؟



انجام عملیات ذخیره‌سازی در این دیدها دارای عوارض بسیاری است که در عمل آنها را ناپذیرا می‌کند.



عملیات ذخیره‌سازی از دید تعریف شده روی چند جدول مبنا (ادامه)

بخش ششم: معماری پایگاه داده‌ها

۵۱

□ دید حاصل از اجتماع، اشتراک، و تفاضل

□ این دیدها از لحاظ تئوری مشکلی در عملیات ذخیره‌سازی ندارند، هرچند نظرات مختلفی مطرح است.

عمل دید	درج	حذف	بهنگام‌سازی
$R_1 \cup R_2$	درج تاپل در R_1 و/یا R_2	حذف تاپل از R_1 و R_2	بهنگام‌سازی تاپل در R_1 و R_2
$R_1 \cap R_2$	درج تاپل در R_1 و R_2	حذف تاپل از R_1 و/یا R_2	بهنگام‌سازی تاپل در R_1 و R_2
$R_1 - R_2$	درج تاپل در R_1 (به شرط عدم وجود در R_2)	حذف تاپل در R_1	بهنگام‌سازی تاپل در R_1 (به شرط عدم وجود در R_2)



□ موضوع دیدهای پذیرا در SQL استاندارد چندان روشن نیست. در SQL 2003 دیدهایی که تمام شرایط زیر را داشته باشند، قابل بهنگام‌سازی (درج، حذف و بروزرسانی) هستند.

[توجه: ممکن است برخی دیگر از دیدها هم از لحاظ تئوری قابل بهنگام‌سازی باشند.]

۱- عبارت تعریف‌کننده دید، یک عبارت SELECT ساده باشد (یعنی شامل عملگرهای UNION، JOIN، INTERSECT و EXCEPT نباشد).

۲- در عبارت SELECT گزینه DISTINCT وجود نداشته باشد.

۳- در کلاز FROM عبارت SELECT، فقط یک جدول وجود داشته باشد.

۴- جدول قید شده در کلاز FROM، یک جدول مبنا یا یک دید قابل بهنگام‌سازی باشد.



۵- در لیست نام ستون‌ها در عبارت `SELECT`، ستون‌های موردنظر باید در جدول مبنا متناظر داشته باشند و به یک ستون از جدول مبنا بیش از یک بار ارجاع وجود نداشته باشد. ضمناً حاوی ستون کلید باشد.

۶- در عبارت `SELECT`، کلاز `GROUP BY` و/یا کلاز `HAVING` وجود نداشته باشد.

۷- کلاز `WHERE` در عبارت `SELECT` حاوی کلاز `FROM` نباشد به گونه‌ای که در آن به همان

جدولی ارجاع داده شده باشد که در کلاز `FROM` ذکر شده در شرط ۴.

نتیجه اینکه عملاً دیدهایی که یک زیرمجموعه افقی-عمودی دارای کلید از یک جدول مبنا (یا از دید قابل بهنگام‌سازی) باشند، قطعاً قابل بهنگام‌سازی هستند.

[توجه: به شرط رعایت محدودیت‌های جامعیتی مانند یکتایی کلید و هیچمقدارناپذیری]

تکنیک دید ذخیره شده [ساخته شده] (Materialized View)

در این تکنیک، دید در سیستم ذخیره می‌شود؛ یعنی دیگر مجازی نیست و جدول ذخیره شده است. تا در هر بار مراجعه به دید لازم نباشد تبدیل E/C انجام شود.

هدف: برای افزایش سرعت عملیات بازیابی.

شرط استفاده: در عمل از این تکنیک وقتی استفاده می‌کنیم که داده‌های ذخیره شده در جدول‌های مبنای زیرین حتی‌الامکان تغییر نکنند. به بیان دیگر، نرخ عملیات ذخیره‌سازی در جدول‌های زیرین پایین باشد. زیرا اگر جدول‌های زیرین تغییر کنند، تغییرات متناسباً در جدول‌های دید باید اعمال شوند و این خود سربار ایجاد می‌کند.

کاربرد: در برنامه‌های آماری، گزارش‌گیری‌ها و برنامه‌های داده‌کاوی (Data Mining)

دید ذخیره شده در SQL چگونه پیاده‌سازی می‌شود؟



برخورد با دید ذخیره‌شده در سمپادهای مختلف متفاوت است. ☐

در PostgreSQL ☐

تعریف دید ذخیره شده مشابه تعریف دید معمولی است. ☐

```
CREATE MATERIALIZED VIEW view-name
AS SELECT .....
FROM ....
```

در بدو تعریف، سیستم داده‌ها را از جداول مبنا استخراج و در دید ذخیره شده ثبت می‌نماید. ☐


با تغییر داده‌های جداول مبنا، داده‌های موجود در دید ذخیره شده به طور خودکار به روز نمی‌شوند و برای به‌روزرسانی نیاز به اجرای دستور زیر است. ☐

```
REFRESH MATERIALIZED VIEW view-name
```



معایب مفهوم دید:

 محدودیت [و مشکلات] در عملیات ذخیره‌سازی

 فزونکاری (overhead) برای انجام تبدیل E/C (محاسبه دید). راه حل: استفاده از تکنیک دید ذخیره

شده



□ مزایای مفهوم دید خارجی:

□ فراهم‌کننده محیط انتزاعی فرافایلی برای کاربران با پویایی بالا

□ اشتراک داده‌ها (Data Sharing) ← داده‌ها یک بار ذخیره می‌شوند و کاربران بسته به نیاز خود از داده‌های ذخیره شده به صورت هم‌روند استفاده می‌کنند.

□ تامین امنیت برای داده‌های زیرین. ← از طریق مفهوم داده مخفی (Hidden Data)، زیرا کاربر خارج از محدوده دید خود هیچ نمی‌بیند (داده‌های نهان تا حدی امن هستند).

□ تامین‌کننده استقلال داده‌ای (مفهوم اساسی در تکنولوژی DB؛ هم مزیت و هم از اهداف مهم تکنولوژی DB).

□ امکانی است برای کوتاه‌نویسی یا ماکرونویسی پرسش‌ها.



چه زمانی از مفهوم دید استفاده نمی‌کنیم؟

□ هنگامی که سیستم تک‌کاربره باشد.

□ هنگامی که به تشخیص admin برای افزایش کارایی سیستم، برخی برنامه‌ها را مستقیماً روی شمای

ادراکی (جداول مبنایی) بنویسیم.

□ هنگامی که کاربر نیازمند انجام عملیات ذخیره‌سازی باشد و از طریق دید امکان آن وجود نداشته باشد.



□ مفهوم استقلال داده‌ای [DI] (جدایی برنامه‌ها از داده‌ها):

□ مصونیت (تاثیرناپذیری) برنامه‌های کاربران [در سطح خارجی] در قبال تغییرات در سطوح زیرین معماری DB.

□ چرا نباید برنامه‌ها تغییر کنند؟

□ چون هر تغییر در برنامه‌ها، هزینه تولید و پشتیبانی و بازتولید برنامه‌ها را بالا می‌برد.

استقلال داده‌ای فیزیکی (PDI)

استقلال داده‌ای منطقی (LDI)

□ استقلال داده‌ای (DI)



استقلال داده‌ای فیزیکی (PDI) ☐

☐ مصونیت برنامه‌های کاربران در قبال تغییرات در شمای داخلی DB

☐ تغییرات در شمای داخلی شامل تغییر در جنبه‌های فایلینگ پایگاه

▪ ساختار فایل، طول رکورد، طرز ذخیره‌سازی فایل روی دیسک، گاه با دخالت طراح فیزیکی و گاه فقط توسط

.DBMS

☐ استقلال داده‌ای فیزیکی در پایگاه داده‌ها به طور کامل تامین است: زیرا کاربران با مفهوم دید کار

می‌کنند که اساساً در سطح فرافایلی مطرح است و برنامه‌ها درگیر جنبه‌های فایلینگ نیستند.



استقلال داده‌ای منطقی (LDI)

مصونیت برنامه‌های کاربران در قبال تغییرات در شمای ادراکی DB.

در سیستم‌های پایگاهی تا حد زیادی این استقلال تامین است ولی نه صددرصد.

تغییر در شمای ادراکی

- رشد پایگاه داده‌ها (DB Growth)
- تغییر سازمان پایگاه داده‌ها [سازماندهی مجدد DB] (DB Restructuring)

نکته: تغییراتی که مورد بررسی قرار می‌دهیم، تغییراتی است که از داده‌ها و ساختار موجود **نمی‌کاهد**،

چرا که تغییرات کاهش، قطعاً بر روی برنامه‌های سطح خارجی تاثیر می‌گذارد و استقلال داده‌ای حفظ نمی‌شود.



❑ چرا رشد DB: مطرح شدن نیازهای جدید

❑ اضافه شدن ستون(های) جدید به جدول(ها)

❑ ایجاد جدول‌های جدید

❑ استقلال داده‌ای منطقی (LDI) در قبال رشد DB، به کمک مفهوم دید تقریباً صددرصد تامین است، زیرا

کاربر دارای یک دید، خارج از محدوده آن دید هیچ نمی‌بیند.



استقلال داده‌ای منطقی (ادامه)

۶۳

بخش ششم: معماری پایگاه داده‌ها



دیده‌های پیش‌تر تعریف شده را روی جدول STT در نظر می‌گیریم. حال نیاز جدیدی برای کاربر مطرح شده است.

V1	
STID	STNAME
کاربر ۱	
:	:

V2	
:	:

V2		
v9	STID	STADR
	:	:

کاربر ۲

STT	STID	STNAME	STL	STJ	STD	STADR
	777	st7	bs	phys	d11	
	888	st8	ms	math	d12	
	444	st4	bs	comp	d14	
	:	:	:	:	:	

ALTER TABLE STT

ADD COLUMN STADR CHAR(70) این گسترش در سطح فایلینگ چگونه انجام می‌شود؟



□ آیا پیرو نیاز جدید کاربر در حد ستون، طراح همیشه جدول مبنا را گسترش می‌دهد؟

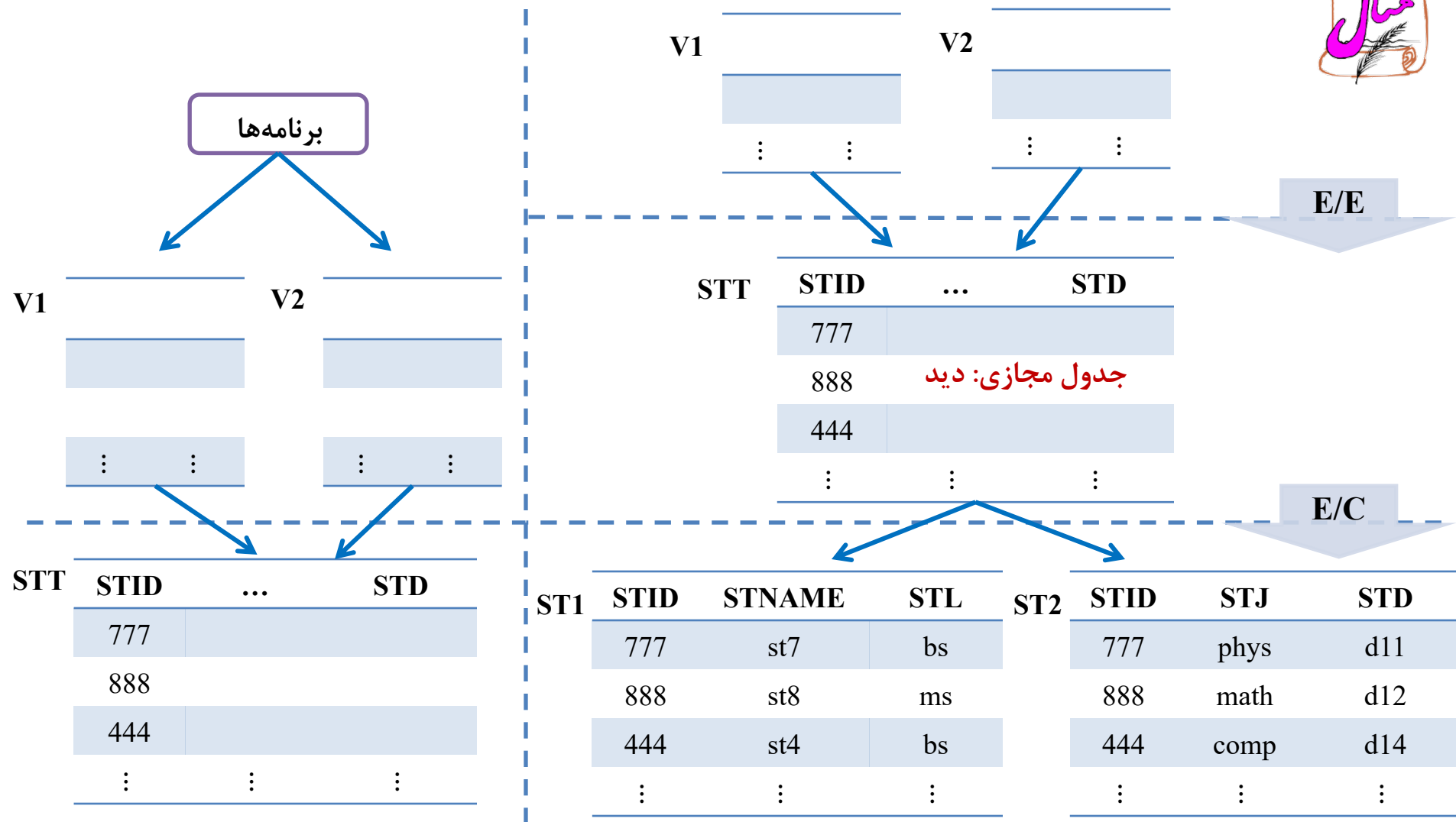
□ خیر، زیرا ممکن است آن ستون مجازی (محاسبه شدنی) باشد.

□ **سازماندهی مجدد DB** یعنی طراح به هر دلیلی طراحی منطقی DB را تغییر دهد. مثلاً یک جدول مبنای

موجود را به دو جدول، تجزیه عمودی کند و طبعاً شمای ادراکی هم تغییر می‌کند. می‌خواهیم ببینیم LDI در قبال این تغییر تا چه حد تامین است.

□ در این حالت، LDI به کمک مفهوم دید و امکان تعریف **دید روی دید** (View Definition on View)،

تا حدی تامین است.





```
CREATE TABLE ST1
```

```
(STID ...,
```

```
...
```

```
STL ...)
```

```
PRIMARY KEY STID
```

شمای جدید:



البته در عمل مشکلات دیگری هم وجود دارد



و صرفاً با این اعمال مشکل برطرف نمی‌شود.

```
CREATE TABLE ST2
```

```
(STID ...,
```

```
...
```

```
STD ...)
```

```
PRIMARY KEY STID
```

```
INSERT INTO ST1
```

```
(SELECT STID, STNAME, STL  
FROM STT)
```

```
INSERT INTO ST2
```

```
(SELECT STID, ..., STD  
FROM STT)
```

مهاجرت داده‌ها (Data Migration)



□ دلایل این نوع تجزیه (که کلید در هر دو جدول باشد) چه می‌تواند باشد؟

□ افزایش کارایی سیستم در رده فایلینگ با فرض 1-Table:1-File برای بعض برنامه‌ها (مثلاً برنامه‌هایی

با فرکانس بالاتری نسبت به ستون‌های ST1 و با فرکانس پایین‌تری به ستون‌های ST2 ارجاع داشته

باشد، فایل‌ها را جدا می‌کند).

□ توزیع داده‌ها در سایت‌ها وقتی پایگاه داده توزیع شده (DDB) داشته باشیم.

□ کاهش حجم Null Value

□ بهینه‌سازی طراحی (رجوع شود به بحث نرمال‌سازی رابطه‌ها)

□ ...



□ با حذف جدول مبنای STT، دیدهای قبلاً تعریف شده روی آن نامعتبر می‌شوند و در نتیجه برنامه‌هایی که روی آنها کار می‌کردند، دیگر اجرا نمی‌شوند و LDI دیگر تامین نیست مگر اینکه طراح و پیاده‌ساز تدبیری

بیندیشد.



✓ جدول STT را با همان نام و ساختار به شکل یک دید تعریف می‌کنیم، با مکانیزم پیوند (دید روی دید):

```
CREATE VIEW STT
AS SELECT STID, ..., STD
FROM ST1 JOIN ST2
```

```
DROP TABLE STT
```

□ تعریف این دید وارد کاتالوگ سیستم می‌شود. ← دیدهای قبلاً تعریف شده معتبر می‌شوند (البته

در عمل ممکن است با مشکلاتی جزئی مواجه گردیم).



□ با این تدبیر، LDI برای برنامه‌هایی که بازیابی انجام می‌دهند، صد درصد تامین می‌شود، به قیمت افزایش سر بار برای انجام تبدیل E/E علاوه بر E/C و C/I. زیرا از تکنیک **دید روی دید** استفاده کرده‌ایم.

□ اما LDI برای برنامه‌هایی که عملیات ذخیره‌سازی انجام می‌دادند، ممکن است تامین نباشد. زیرا این بار STT خود یک دید است و دیده‌ها در عملیات ذخیره‌سازی عمدتاً مشکل دارند. در مثال ارایه شده نیز از نظر **تئوریک** مشکلی بروز نمی‌کند ولی در **عمل** ممکن نیست. ← چون STT یک دید پیوندی PK-PK است.



پرسش و پاسخ ...

amini@sharif.edu