فهرست

۱۹,۱. مقدمه.

اغلب اطلاعات در جهان واقعی یا از دسترس کاربران خارج می‌شوند یا غیر قابل اعمال و یا تعریف نشده هستند؛ برای مثال :

- تاریخ تولد فرد مورد نظر مشخص نیست.
- هویت گوینده‌ای که وظیفه خبر رسانی به صورت صوتی را دارد موجود نیست.
- آدرس جدید افرادی که قبلاً در محلی زندگی می‌کردند در دسترس نیست.

و مواردی از این قبیل که برای ما نا آشنا هم نیستند، جزو چنین اطلاعاتی محسوب می‌شوند. بنابراین به راهکاری نیاز داریم تا سیستم‌های پایگاه داده، با چنین اطلاعاتی درست کار کنند. زبان SQL بدین منظور در عمل روش‌هایی به کار گرفته است. این زبان در بسیاری از تولیدات تجاری بر اساس منطق سه ارزشی[1] (3VL) و هیچ مقدارپذیرها بنیان شده است.

برای مثال، ممکن است که ما از وزن دقیق برخی قطعات مطلع نباشیم در این صورت part یا Loosely را گزارش می‌کنیم. بدین معنا که وزن قطعه‌ی مذکور تهی شده یعنی اینکه؛

الف- از وجود چنین قطعه‌ای اطلاع داریم و البته، ب- می‌دانیم که وزن هم دارد اما، اطلاعی از وزن قطعه وجود ندارد.

[1] Three valued logic

می‌خواهیم بیشتر روی این مسأله دقت کنیم، تاپلی را که part P_7 را در پایگاه داده نشان می‌دهد را در نظر بگیرید.

بدیهی ست که نمی‌توانیم ارزش حقیقی وزن (WEIGHT) را در تاپل قرار دهیم. کاری که می‌توانیم انجام دهیم این است که موقعیت WEIGHT را به عنوان تهی با پرچم [1]یا علامت [2]نشانه گذاری کرده و در تاپل قرار دهیم. سپس علامت یا پرچم را تعریف می‌کنیم، بدین صورت که ما هیچ اطلاع و صفتی از ارزش حقیقی نداریم.

در این مرحله به این موضوع فکر می‌کنیم که موقعیت WEIGHT به عنوان تهی [3]باقی بماند یا ارزش آن به تهی تبدیل شود؛ و البته در مورد تمام این مسائل به صورت عملی بحث کنیم.

اما بایستی به این موضوع توجه شود که این گفتگوها تنها به صورت غیر رسمی بوده و چندان هم دقیق و بدون خطا نیستند. با تمام این اوصاف، سازند و مؤلفه‌ی WEIGHT در بعضی از تاپل‌ها به صورت هیچ مقدار است یعنی is null. بنابراین تاپل هیچ ارزش وزنی‌ای برای قطعه قائل نیست. پس جای سوال است که عبارت هیچ مقدار پذیر یا تهی value که بارها آن‌ها را شنیده‌ایم، عبارت ناملموسی به حساب می‌آیند. قضیه از این قرار است که هیچ مقدارپذیرها، خود ارزش نیستند؛ آن‌ها تکرار کار، و نشانه‌هایی مانند علامت و پرچم در پایگاه داده‌ها هستند. در قسمت‌های بعد خواهید دید که مقایسه‌ی اسکالر در هر کدام از قیاس وندها به جای true و false برای مقادیر ناشناخته، هیچ مقدار پذیر "تهی" ارزیابی می‌شوند.

به طور کلی برای چنین مواردی از تهی به عنوان یک مقدار ناشناخته "value UNKnown" تفسیر می‌کنند. مانند عبارت A > B و بدون در نظر گرفتن ارزش B. به خاطر داشته باشید که دو هیچ مقدار پذیر به این عنوان که یکی مساوی

---

[1] flag
[2] mark
[3] null

دیگری ست محسوب نمی‌شود یعنی مقایسه‌ی A = B، ناشناخته و UNKnown است درست نیست. باز هم زمانی می‌توانیم چنین حرفی بزنیم که هر دو عامل ما تهی باشند؛ و به طبع این موضوع، اعلام UNKnown برای نامساوی بودن A و B نیز نادرست است. با توجه به منطق سه ارزشی، مفهوم تهی حداقل در این واحد، معمولاً قابل فهم است و ضرورتاً ما را به منطقی که در آن جا سه ارزش حقیقی وجود دارد هدایت می‌کند یعنی (True, False, UNKnown) .

قبل از اینکه بیشتر پیش برویم بایستی از هیچ مقدارپذیرها و منطق سه ارزشی که در مشکلات جدی بوده و هستند و جایی نیز در مدل‌های مربوط ندارند هم ما و هم بسیاری از نویسندگان، تصویر روشنی داشته باشیم.

برای مثال، یک تاپل مشخص هیچ ارزش WEIGHT ای را شامل نمی‌شود به عبارت دیگر جزئی از تاپل و، به عنوان نمونه‌ای از یک مسند کاربردی تلقی نمی‌شود.

محتویات مطالبی که در این قسمت و در قسمت‌های بعد از مقدمه خواهد آمد، حس بی اعتمادی در مورد عدم تمایل در بحث تهی s و 3VL را تا مدتی به تعویق می‌اندازد و در این قسمت بدون توجه به انتقادهایی که تا کنون رفته است ما نظرات بنیادین مربوطه به هیچ مقدارپذیرها و منطق سه ارزشی را تعریف خواهیم کرد. برآورد ما از این قضیه صحیح است چرا که تا زمانی که تعریف درستی از نظرات بنیادین نداشته باشیم نمی‌توانیم در مورد آن‌ها قضاوت منصفانه انجام دهیم چه برسد به اینکه قصد انتقاد از آن‌ها را داشته باشیم. در بخش ۱۹,۳ به بحث در مورد برخی از مهم‌ترین سازنده‌ای نظرات خواهیم پرداخت. مخصوصاً در مورد موقعیت ما نسبت به هیچ مقدارپذیرها، که آن را اشتباه می‌پندارند. قسمت ۱۹,٤ نیز به عملیات منطقی کلیدهای خارجی و اصلی رسیدگی می‌کند. قسمت ۱۹,۵، به OUTER JOIN یا همان پیوندهای بیرونی می‌پردازد؛ و با مجموعه عملیاتی که در محتوای تهیs و 3VL هستند رو به رو می‌شویم. قسمت ۱۹,٦ به

روش‌های دیگر اطلاعات نهست با استفاده از تک ارزشی‌ها special values رسیدگی می‌شود. در قسمت ۱۹,۷ ترسیم شمایی از SQL و قسمت ۱۹,۸ خلاصه‌ای از مطالب جمع آوری شده است.

توجه : دلایل زیادی مبنی بر عدم توانایی ما در قرار دادن ارزش اصلی داده‌ها در برخی موقعیت‌ها بین تاپل‌ها وجود دارد. VALUE UNKNOWN یک از این دلیل‌ها است. از دلایل دیگر می‌توان به :

عدم کاربرد پذیری ارزش.

وجود نداشتن ارزش مربوطه.

تعریف نشدن ارزش و تأمین نشدن آن اشاره کرد.

از جمله پیشنهاداتی که در این زمینه ارائه شده است می‌توان به فرضیه‌ی بسط الگوها اشاره کرد که در این صورت هیچ مقدارپذیرها یا همان تهی‌ها، تنها یک معنی را نپذیرند بلکه هم معنی ناشناخته بودن ارزش را داشته باشند و هم کاربرد ناپذیری آن را. پیشنهاد دوم این است که DBMS ها نباید تنها با منطق سه ارزشی سر و کار داشته باشند و باید آن‌ها را به منطق‌های چهار ارزشی بسط دهند. در قسمت ۱۹,۵ به پیشنهادهای دیگر نیز می‌پردازیم.

در این قسمت توجه شما را صرفاً به یک نوع از تهی‌ها جلب می‌کنیم، تهی‌های ناشناخته‌ی مقادیری؛ VALUE UNKNOWN تهی، که برای این تعریف از این به بعد و البته نه همیشه به UNK رجوع می‌کنیم.

۱۹,۲. نظری اجمالی بر روش منطق سه ارزشی 3VL

در این قسمت به اختصار در مورد مؤلفه‌ها و سازند های منطق سه ارزشی در مورد اطلاعات نهست به بحث می‌نشینیم. در آغاز به بررسی اثر هیچ مقدار پذیر با استفاده از UNK ها در عبارات بولین Boolean می‌پردازیم.

اپراتور و عمل وندهای Boolean

در این قسمت با جرات اظهار می‌شود که هرگونه قیاس اسکالر در هر قیاس وندی، ارزیابی UNK بر ارزش‌های حقیقی بی مقدار UNKNOWN TRUTH VALUE به جای TRUEو FALSE می‌باشد و از این بابت ما با منطق سه ارزشی سر و کار خواهیم داشت. UNKnown سومین ارزش حقیقی است. جداول منطق سه ارزشی را برای AND، OR و NOT را در قسمت ذیل می‌بینید.

(t=true, f=false, u=UNK).

AND	**T u f**
T	**T u f**
U	**T u f**
f	**F f f**

OR	**T u f**
T	**T t t**
U	**T u u**
f	**T u f**

NOT	
T	**F**
U	**U**
F	**T**

برای مثال، به A و B به ترتیب مقادیر ۳ و ٤ و برای C مقدار ثابت UNK را فرض کنید. در نتیجه با توجه به عبارات، ارزش هر کدام نشان داده می‌شود؛ برای درک این مطلب به قسمت ذیل توجه فرمایید:

A > B AND B > C: false
A > B OR B > C: unk
A < B OR B < C: true
NOT (A = C)

به عبارت دیگر چنانچه:

الف) r رابطه‌ی بین تاپل‌های t1, t2,…,tm باشند.

ب) V متغیر طیفی‌ای باشد که طیف r را پوشش دهد و p(V) عبارت بولینی با شد که به ازای هر V به عنوان متغیر آزاد رخ دهد در آن صورت خواهیم داشت:

EXISTS V (p) V

که مساوی رابطه‌ی زیر خواهد بود :

False OR p (t1) OR … OR p (tm).

همچنین عبارت زیر نیز:

FORALL V (p) V

برابر خواهد بود با :

True AND p (t1) AND … AND p (tm).

بنابراین چنانچه p(ti) برای UNK ارزیابی می‌شد به ازای هر i چه اتفاقی می‌افتاد؟

در آن صورت با وجود رابطه r دقیقاً تاپل‌های زیر را خواهیم داشت:

(1, 2, 3)

(1, 2, UNK)
(UNK, UNK, UNK)

برای درک راحت‌تر مسئله، فرض کنید سه صفت یا به عبارتی سه خصیصه، از چپ به راست همان‌طور که نشان داده شده است، A، B و C نامیده شوند. هر کدام از آن‌ها نیز یک عدد صحیح به حساب می‌آیند.

با تمام این شرایط عبارات زیر، ارزش‌هایی را خواهند داشت که در جلوی آن‌ها منظور می‌شوند.

EXISTS V (V.C > 1): TRUE
EXISTS V (V.B >2): UNK
EXISTS V (MAYBE (V.A> 3)): TRUE
EXISTS V (IS_UNK (V.C)): TRUE
FORALL V (V.A>1): FALSE
FORALL V (V.B >1): UNK
FORALL V (MAYBE (V.C>1)): FALSE

اپراتورهای دیگر سنجش.

عبارت عددی زیر را در نظر بگیرید :

WEIGHT * 454

جایی که WEIGHT وزن چیزی را نمایش می‌دهد، اگر به عنوان UNK در عبارت باشد چه اتفاقی خواهد افتاد؟ ارزش عبارت چه خواهد شد؟ پاسخ این است که اگر سایر عمل وندهای عبارتی خودشان UNK باشند، آن عبارت عددی در UNK ارزیابی می‌شوند. بنابراین برای مثال اگر WEIGHT به عنوان UNK باشد آنگاه تمام عبارت زیر با UNK ارزیابی می‌شوند:

WEIGHT + 454	454 + WIEGHT	+WEIGHT
WEIGHT - 454	454 - WIEGHT	-WEIGHT
WEIGHT * 454	454 * WIEGHT	
WEIGHT / 454	454 / WIEGHT	

شاید لازم به توضیح باشد که عملکرد قبلی ما از عبارات عددی موجب بعضی نابسامانی‌ها خواهد شد. برای مثال عبارت WEIGHT-WEIGHT که باید برابر ارزش صفر باشد، در حقیقت UNK را به ما می‌دهد؛ و در عبارت WEIGHT/0 که باید پیام ZERO divide را داشته باشد هم UNK جواب خواهد بود. (در هر دو حالت مقدار اولیه‌ی UNK, WEIGHT می‌باشد). ما فعلاً چنین مغایرت‌هایی را تا قسمت‌های بعدی نادیده در نظر می‌گیریم.

مشاهدات مشابهی برای عمل وندها و سنجش‌های دیگر تقاضا می‌شوند بجز در موارد ذیل:

الف اپراتورهای Boolean

ب اپراتور IS_UNK

ج اپراتور IF_UNK

برای مثال، رشته‌ی کاراکتری عبارت A || B به UNK مراجعت داده می‌شود اگر A یا B یا هر دوی آن‌ها UNK باشند.

اپراتور IF_UNK دو عمل وند عددی را به خود می‌گیرد و به مقدار اولین عمل وند مراجعت می‌کند مگر اینکه عمل وند به UNK ارزیابی کند. در چنین حالتی مقدار دومین عمل وند را در نظر خواهد گرفت (به عبارت دیگر اپراتور راهی را برای تبدیل

یک UNK به بعضی مقادیر بدون UNK فراهم می‌کند). برای مثال فرض کنید UNK ها برای تولید کنندگان خصیصه‌ی CITY مجاز شده‌اند، بنابراین عبارت:

EXTEND S ADD IF_UNK (CITY,"CITY UNKNOWN") AS SCITY

برای هر تولید کننده‌ای که CIY به عنوان UNK در S به آن‌ها داده شده CITY UNKNOWN را به عنوان جواب به اطلاع می‌رساند.

ضمناً آن IF_UNK می‌تواند بر حسب IS_UNK ثبت شود:

IF_UNK (exp1, exp2)

(در اینجا exp2 , exp1 باید از یک نوع باشند) بنابراین عبارت بالا به صورت زیر تعریف خواهد شد:

IF IS_UNK (EXP1) THEN EXP2 ELSE EXP1 END IF

UNK، UNK نیست.

مهم است بدانید : UNK مخفف عبارت the Value-UNKnown تهی به معنای مقدار ناشناخته‌ی نال و UNK مخفف عبارت the UNKnown truth value می‌باشد. پس بایستی این دو را نسبت به یکدیگر تمیز داد. مهم‌ترین تمایزی که بین این دو وجود دارد این است که UNK خود، یک ارزش و یک مقدار است در صورتی که UNK به هیچ عنوان، ارزش محسوب نمی‌شود. اما اجازه بدهید تا کمی تخصصی‌تر به این مسأله توجه کنیم. فرض کنید X، متغیر نوع بولین (Boolean) باشد. X باید یکی از ارزش‌های true، false یا UNK را داشته باشد. چنانچه وضعیت UNK برای x انتخاب

شود در این صورت ارزش x برای UNK بودن شناخته شده است. بالعکس این قضیه نیز صدق می‌کند؛ اگر X، UNK باشد در آن صورت ارزش آن ناشناخته خواهد بود.

آیا انواع (type) نیز جزو UNK محسوب می‌شود؟

حقیقت این است که خود UNK به عنوان یک ارزش و یک مقدار تلقی نمی‌شود به همین جهت هیچ نوعی نیز نمی‌تواند UNK باشد چرا که انواع خود مجموعه‌ای از مقادیر هستند.

اپراتورهای رابطه‌ای.

در این قسمت توجه شما را به اثرات UNK ها روی اپراتورهای جبر رابطه‌ای معطوف می‌کنیم.

برای درک بهتر حوزه‌ی عملیاتی خود را به اپراتورهای تولید، گزینش، پرتو گیری، اجتماع و تمایزات محدود می‌کنیم البته اثر UNK ها بر سایر اپراتورها از اثراتشان بر این پنج مورد سرچشمه می‌گیرد.

اولین مورد product و تولید است که ساختاری ساده در این حوزه می‌باشد.

دومین مورد، اپراتوری گزینشی ست که تا حدودی به رابطه‌ای گرویده که شامل تنها تاپل‌هایی می‌شود که شرایط گزینش را true ارزیابی می‌کنند، نه false و نه UNK .

لازم به توجه است که ما در نمونه‌های خود این تغییر کارایی را در قسمت اپراتورهای بولین قبلاً لحاظ کرده‌ایم.

مورد سوم projection یا پرتو گیری ست که در این اپراتور تاپل‌های تکثیر شده‌ی افزونه، حذف می‌شوند.

در منطق دو ارزشی سنتی (conventional two-valued logic) یا به عبارتی 2VL سنتی، دو تاپل t_1 و t_2 کپی یکدیگر خواهند بود اگر و تنها اگر هر دو تنها یک تاپل عین هم باشند. به عبارت دیگر تجزیه‌ی مقادیری آن به مقادیر یکسان منجر شود.

برای درک بهتر می‌توانیم چنین مثالی برای شما بیاوریم که دو تاپل را می‌توان نسخه‌ای برابر با دیگری فرض کرد اگر و تنها اگر :

چنانچه تاپل ما از اجزای A1,A2, … , An تشکیل شده باشد در آن صورت به ازای.

i = 1, 2, 3, …, n، مقدار Ai در t_1 برابر مقدار Ai در t_2 باشد.

البته در منطق سه ارزشی مقادیر برخی از متغیرها و صفات ممکن است برابر UNK شود که در این صورت شما به خوبی می‌دانید که UNK برابر هیچ چیزی نیست حتی خودش.

بنابراین می‌توان به این استنباط برسیم که اگر تاپلی شامل UNK بود، کپی هیچ تاپل دیگری نیست حتی تاپل مشابه خودش؟

ممکن است شما به سرعت پاسخ مثبت به این سوال بدهید در صورتی که چنین نیست. با این وجود که UNK برابر هیچ چیزی نیست حتی خودش اما در قضیه‌ی دوبل و کپی، هدف این است یک چیزی برابر یک چیز دیگری باشد خواه مقدار داشته باشد خواه نداشته باشد.

دلیل چنین مغایرتی ذیلاً نشان داده شده است:

تست انتقال و جا بجایی در این مورد؛ در درجات پایین‌تر از آزمایش جا بجایی در ارزیابی شرایط بازیابی قرار می‌گیرد:

امکان چنین تفاوت‌هایی در نقوش خصیصه‌ها وجود دارد.

در این مورد که آیا این عبارت صحیح و منطقی ست، بحث نمی‌کنیم و قضاوت را به عهده‌ی شما می‌سپاریم تا ببینید آیا چنین رابطه‌ای قابل قبول هست یا خیر؟. با این وجود اجازه دهید تا آن را فعلاً قبول داشته باشیم. با قبول چنین شرایطی تعریفات زیر را نیز باید بپذیریم:

- دو تاپل t1 و t2 کپی یکدیگر خواهند بود اگر و تنها اگر هر دو تنها یک تاپل عین هم باشند.

به عبارت دیگر چنانچه تاپل ما از اجزای A1,A2, … , An تشکیل شده باشد در آن صورت به ازای i = 1, 2, 3, …, n، مقدار Ai در t1 برابر مقدار Ai در t2 باشد. حتی اگر Ai در t1 و Ai در t2 هر دو UNK باشند.

با وجود چنین تعریفی، از تاپل‌های دوبل، تعریف اصلی اپراتور پرتو گیری تغییر نمی‌کند.

- t1 = t2
- t1و t2 هم مقدار یکدیگرند.

قضیه‌ی اتحاد (union) نیز عقیده‌ای به همین منوال داد بدین معنا که در پی حذف تاپل‌های دوبل افزونه می‌باشد. بنابراین ما اجتماع دو رابطه‌ی r1 و r2 را که از یک نوع هستند را تعریف می‌کنیم. در این دو رابطه ساختار r شامل تمام تاپل‌های ممکنه است که t خصیصه‌ای در بعضی از تاپل‌های t1است که در برخی تاپل‌های r2 نیز تکرار شده‌اند.

مگر اینکه حذف تاپلی در این بین رخ ندهد که در این صورت بایستی تفاوت‌ها به صورت تمایزی تعریف شوند:

برای مثال: تاپل t در r1 از r2 کمتر خواهد بود اگر و تنها اگر کپی بعضی از تاپل‌های خود r1 باشد نه جزو تاپل‌های r2.

به روز درآوری اپراتورها.

دو نکته‌ی بسیار مهم در به روز درآوری اپراتورها وجود دارد.

۱- چنانچه صفت A از B اجازه‌ی UNK را بدهد و عاملی نیز از طریق INSERT به منظور جای نهادن تاپلی در R بدون در نظر گرفتن ارزشی برای A، ساخته شود. در آن صورت سیستم به صورت خودکار، UNK را در موقعیت A در

تاپل قرار خواهد داد. در این مرحله ما چنین فرض کرده‌ایم که هیچ نقص UNK ای برای ارزش و مقدار A تعریف نشده است. چنانچه صفت A از رلوار B (relvar B) مجوز ارزش UNK را ندهد، عاملی نیز برای قرار دادن تاپلی در R با INSERT یا UPDATE ساخته می‌شود که در آن موقعیت یعنی در موقعیت A، UNK به عنوان ERROR شناخته می‌شود.

۲- عاملی که باعث کپی سازی تاپل به کمک INSERT و UPDATE در R می‌شود نیز خطا منظور می‌گردد. در این قسمت هم معنای تاپل‌های افزونه به همان صورت قبلی تعریف می‌گردد.

جامعیت محدودیت.

همان‌طور که در فصل ۱۹ توضیح داده شد، ممکن است کمتر به جامعیت محدودیت به عنوان عبارت بولین که نباید FALSE ارزیابی شود توجه شود. تذکر: محدودیت چنانچه به عنوان UNK ارزیابی شد نباید به منظور تخلف در قانون داده‌ها رسیدگی شود. در مواردی این‌چنینی باید بگوییم که مقدار، ناشناخته و تعریف نشده است چه بسا محدودیت در آن اعمال شده باشد.

۱۹،۳. بعضی از نتایج طرح‌های قبلی.

روش منطق ۳ ارزشی که در قسمت قبل توضیح داده شد، چند نتیجه‌ی منطقی دارد که همه‌ی آن‌ها الزاماً معلوم نیستند. در این قسمت بعضی از نتایج حاصل از طرح‌های قبلی را توضیح داده و در مورد اهمیت آن‌ها بحث می‌کنیم.

تبدیل عبارات.

ابتدا توجه داشته باشید که در این قسمت عباراتی که در منطق ۲ ارزشی true ارزیابی می‌شدند، در منطق ۳ ارزشی الزاماً به این صورت ارزیابی نخواهند شد.

- برابری x = x همیشه پیام true را برای شما نخواهد داشت.

در منطق ۲ ارزشی (2VL)، x همواره برابر خودش است. در عین حال در منطق سه ارزشی (3VL) در صورتی که x، UNK باشد، چنین برابری را نخواهیم داشت.

- عبارت بولینی p OR NOT (p) الزاماً پیام true را برای شما نخواهد داشت.

در منطق دو ارزشی عبارت p OR NOT (p) زمانی که p یک عبارت بولینی باشد همواره true ارزیابی خواهد شد. در عین حال در منطق ۳ ارزشی، چنانچه p، UNK ارزیابی شود در آن صورت کل عبارت UNK OR NOT (UNK) مقدار گذاری و ارزیابی خواهد شد.

اگر ما رویه‌ی سوال «تمام تأمین کنندگان در شهر لندن» را فعال کنیم در آن صورت ما در این حالت سوال دیگری را نیز پرسیده‌ایم به این صورت که «تمام تأمین کنندگانی را که در لندن نیستند» را نیز می‌خواهیم؛ و در نتیجه، تمام اطلاعات در مورد این دو سوال کسب خواهیم کرد. البته ما لزوماً تمام تأمین کنندگان را نمی‌خواهیم. در عوض اطلاعات ما باید طوری باشد که تمام تأمین کنندگانی که ممکن است در لندن هم نباشند را لازم داشته باشیم. به بیان دیگر ارزش آن‌هایی که true ارزیابی می‌شدند در آنالوگ 2VL در عبارتی مانند p OR NOT (p) به صورت p OR NOT (p) OR MAYBE (p) خواهند بود.

بهتر است بیشتر روی مسئله و مثال قبلی تأمل کنیم. کلید اصلی این است که : از آنجا که در دو مورد؛ " city is London" و "city is not London"، مجموعه‌ی وسیعی از احتمالات در جهان واقعی شامل نمی‌شود، پایگاه داده‌ها جهان واقعی را ملاک قرار نمی‌دهد. در عوض، تنها اطلاعات و علوم موجود و واقعی مربوط به دنیای حقیقی را در پایگاه داده‌ها قرار داده می‌شوند.

بنابراین در مورد دنیای حقیقی ما سه مورد احتمالی خواهیم داشت نه دو احتمال:

این سه احتمال را به صورت مثالی برای شما می‌آوریم؛

" city is known to be London : شهر به نام شهر لندن شناخته شده است « »
city is know not to be London: شهر به نام شهر لندن شناخته نشده است «و»
city is not known : شهر ناشناخته است «. گذشته از این البته ما به صورت مستقیم نمی‌توانیم از سیستم سوال‌ها در مورد دنیای حقیقی سوال کنیم، ما تنها از آن می‌توانیم در مورد علومی از دنیای واقعی سوالاتی کنیم که قبلاً به صورت داده در پایگاه داده ثبت شده باشد. کاربر بر حسب ناحیه‌هایی که در دنیای واقعی هست فکر می‌کند اما سیستم بر اساس اطلاعات خودش از دنیای واقعی پاسخ می‌دهد.».

از آنچه تاکنون نوشتیم چنین بر می‌آید که چنین سردرگمی‌ای محدودیت و تله‌ای ست که هر کسی ممکن است در آن بیفتد اما با این حال به خاطر داشته باشید که تمام مثال‌ها و تمرینات و تمام مواردی که در این سری از مطالعات آمده است تماماً در مورد خود دنیای حقیقی ست نه علوم و اطلاعات بسته بندی شده‌ای که در پایگاه داده‌ها درباره‌ی دنیای حقیقی وجود دارد.

- مقدار عبارت r JOIN r همیشه به شما مقدار r را نخواهد داد.

در منطق دو ارزشی، شکل گیری پیوند بین رابطه r با خودش همواره برابر عبارت و مقدار اصلی‌اش یعنی r می‌باشد. در عین حال در منطق سه ارزشی تاپل UNK در هر پرسشی با خودش پیوند برقرار نمی‌کند زیرا پیوند (join) برخلاف اتحاد و اجتماع در قسمت قبل (union) بر اساس فضای بازیابی و retrieval style در آزمایشات برابری نهادینه شده نه فضای المثنی و دو نسخه‌ای(duplicate – style).

- INTERSECT دیگر مورد ویژه‌ای برای پیوند و JOIN به حساب نمی‌آید. این یکی ارز همان نتایجی ست که قبلاً ذکر کردیم. این پیامد ناشی از retrieval style یا حالت بازیابی پیوند در آزمایش‌های برابری و قیاسی ست در حال

یکه تابع نهایی حاصله از دو ورودی صحیح یا همان INTERSECTION بر اساس قاعده‌ی المثنی ست.

- چنانچه داشته باشیم A = B و B = C در آن صورت نمی‌توان نتیجه گرفت که A = C به عبارت دیگر رابطه‌ی تعدی بر قرار نیست.

بر اساس روابط فوق، روابط و معادلاتی که در منطق دو ارزشی 2VL موجود هستند در 3VL و منطق سه ارزی دیگر کارایی خاصی نخواهند داشت و بنیان آن‌ها به هم خواهد ریخت. یکی از جدی‌ترین نتایج چنین تجزیه‌ی ساختاری‌ای به این شکل است که؛ به طور کلی معادلات و مساوی‌های ساده‌ای مانند r JOIN r = r در قلب قوانین انتقالات متعددی قرار می‌گیرد که برای تبدیل سوالات به برخی فرم‌های کارآمد به کار می‌روند، همان‌طور که در فصل ۱۸ نیز در این مورد توضیح داده شد. علاوه به راین قوانین مذکور نه تنها توسط سیستم به هنگام بهینه سازی بلکه توسط کاربران به هنگام پیدا کردن بهترین راه برای وضعیت سوال داده شده به کار می‌روند؛ و چنانچه هم ارزی فوق بر قرار و موجود نباشد، قانونی نیز در دسترس نخواهد بود.

عکس این عبارت نیز برقرار است بدین صورت که چنانچه قانونی وجود نداشته باشد آنگاه تغییر و تبدیلی وجود نخواهد داشت، در نتیجه پاسخ‌های اشتباهی را در نتیجه‌ی خروجی سیستم خواهیم داشت.

مثال‌هایی برای Department و Employee که به ترتیب با مخفف‌های DEPT و EMP نشان داده می‌شوند.

علاوه بر اینکه در مورد مشکلات ناشی از تغییر و تبدیلات نادرست صحبت می‌کنیم، مثال‌های مخصوصی را نیز آورده و در مورد آن‌ها بحث خواهیم کرد. (مثال از مرجع قسمت ۱۹,۹ گرفته شده چرا که در این جا ضروری نیست و بجای

اینکه بر اساس روابط جبری باشد مبنی بر حساب رابطه‌ای ست). به دیتا به یس و پایگاه داده‌ی DEPT-EMP و عبارت زیر در تصویر ۱۹,۱ توجه کنید:

DEPT.DEPT# = EMP.DEPT# AND EMP.DEPT# = DEPT# (“D1”).

که شاید ممکن است جزئی از یک سوال باشد. در اینجا DEPT و EMP متغیر طیفی هستند. برای تنها تاپل‌هایی که در پایگاه داده وجود دارند این عبارت مقدار و ارزشی برابر UNK AND UNK دارد. بدین معنی که یک برنامه‌ی تنظیم کننده‌ی خوب با توجه به اینکه فرم‌های a =b و b=c را درک می‌کند با تعمیم این رابطه، عبارت زیر را مبنی بر a=c را در جمله‌ی طیفی گزینش خود می‌افزاید:

DEPT.DEPT# = EMP.DEPT# AND EMP.DEPT# = DEPT# {‘D1’}
AND DEPT.DEPT# = DEPT# {‘D1’}

عبارت اصلاح شده چنین ارزیابی می‌شود : UNK AND UNK AND false، یعنی برای آن دو تاپل موجود در پایگاه داده false مقدار و ارزش نهایی خواهد بود. بنابراین سوالی مانند رابطه‌ی ذیل:

DEPT.DEPT# = EMP.DEPT# AND EMP.DEPT# = DEPT# {‘D1’}
AND DEPT.DEPT# = DEPT# {‘D1’}

به employee E1 بر می‌گردد هرچند در روند جدول ۱۹,۱ صادق باشد و در غیر این صورت عمل نکند.

بنابراین بهینه سازی‌هایی را شاهد خواهیم بود که کاملاً موجود هستند؛ و در منطق دو ارزشی سنتی قابل استفاده و در منطق سه ارزشی کارایی ندارند.

DEPT

DEPT#
D2

EMP

EMP#	DEPT#
E1	UNK

استلزام تفهیم مثال قبلی برای تعمیم یک سیستم با منطق دو ارزشی برای پشتیبانی منطق سه ارزشی :

این کار امکان پذیر است منت‌های مراتب، چنین تعمیمی احتمالاً نیازمند مهندسی سازی سیستم‌های موجود است. زیرا چنین بخشی از کد بهینه ساز موجودیت، احتمالاً غیر فعال شده است. در بدترین حالت، مشکلات عدیده ای ایجاد خواهد شد. از همه بیشتر به تعمیم سیستمی که منطق n ارزشی را پشتیبانی می‌کند توجه کنید تا در عوض منطق n+1 ارزشی را پشتیبانی کند. به ازای هر مقداری که به n افزوده می‌شود به مشکلات الگوریتم کد گذاری ارزش‌های مرتبه‌ی n نیز اضافه می‌شود.

تفسیر.

اجازه دهید تا مثالی را که درباره‌ی DEPT و EMP زدیم را بهتر و با دقت بیشتری بررسی کنیم. از آنجایی که employee E1 در دنیای حقیقی department مشابه خود را دارد، UNK طالب ارزش‌های واقعی است و جواب d می‌دهد. در این حالت d هم D_1 هست و هم نیست. اگر باشد در این صورت مقدار و عبارت نهایی.

DEPT.DEPT# = EMP.DEPT# AND EMP.DEPT# = DEPT# {‘D1’}

به ازای داده‌ی داده شده false ارزیابی می‌شود زیرا جمله‌ی طیفی اول false ارزیابی می‌شود؛ و چنانچه d برابر D1 نباشد، در این صورت نیز عبارت قبل، false ارزیابی می‌شود زیرا دومین قسمت جمله، false ارزیابی شده است. به بیان دیگر عبارت داده شده در دنیای حقیقی همواره false ارزش گذاری می‌شود صرف نظر از اینکه UNK چه ارزشی را خواستار است. بنابراین نتیجه‌ای که برای منطق سه ارزشی صحیح است با نتیجه‌ای که برای دنیای حقیقی قابل قبول است یکسان نیستند! به عبارت دیگر منطق سه ارزشی بر اساس خواسته‌های و علایق دنیای حقیقی رفتار نمی‌کند. به نظر نمی‌رسد که 3VL تفسیر محسوسی بر حسب مشغولیت‌های دنیای واقعی داشته باشد.

توجه: مسائل مربوط به تفسیر مشکلی نیست که از طرف تهی و 3VL باشد (به مرجع [19.1-19.11] توجه کنید). یک مسئله‌ی اساسی نیز نیست اما شاید یک عمل معنا دار باشد.

مسندیات.

فرض کنید رابطه‌ای که ارزش رلوار EMP را شامل می‌شود تنها دارای دو تاپل باشد، (E_1,UNK) و (E_2,D_2) . اولین مورد مشابه گزاره " employee به عنوان E_2 در department، D_2 مشخص شده «و دومین مورد مشابه گزاره[1]» employee ای به عنوان E_1 مشخص شده « . به خاطر داشته باشید که یک تاپل شامل UNK می‌شود بدین معنا که تاپل در حقیقت شامل هیچ چیزی نمی‌شود بنابراین؛ تاپل (E_1,UNK)، القا شده و به عنوان تاپل در این لحظه شناخته نمی‌شود و تنها باید به عنوان E_1 به آن نگاه شود. به عبارت دیگر دو تاپل مذکور نماینده‌ی دو مسند [2]متفاوت هستند و رابطه‌ی به هیچ وجه به این معنی تعریف نمی‌شود اما در عوض نوعی اتصال و نه از نوع رابطه‌ای، با دو عنوان متفاوت ارتباط دارند.».

ممکن است چنین پیشنهاد کنند که این موقعیت با ادعای این موضوع که تنها یک مسند وجود دارد تسهیل یابد. مسندی که شامل OR می‌شود، مانند رابطه‌ی زیر:

There is an employee identified as E# in the department identified as D# OR there is an employee identified as E#.

با تمام این اوصاف، روابط باید شامل تاپلی با فرم (E_i,UNK)، به ازای هر employee E_i باشد. تعمیم چنین راه حلی برای رابطه‌ای که شامل UNK ها در چندین صفت‌های مختلف می‌شود، تفکری تقریباً ترسناک است. (برای مواردی رابطه در نهایت هنوز رابطه نیست پاراگراف بعدی را بخوانید).

[1] proposition
[2] predicate

چنانچه ارزش صفت داده شده در تاپل و رابطه‌ی داده شده، UNK ارزیابی شود در آن صورت موقعیت آن صفت در حقیقت شامل هیچ چیز نمی‌شود و بر این مهم دلالت می‌کند که صفت داده شده، یک صفت و خصیصه به حساب نمی‌آید. تاپل، تاپل نخواهد بود، رابطه، رابطه نیست و شالوده‌ی آنچه ما انجام می‌دهیم دیگر بر اساس تئوری روابط ریاضی نخواهد بود. به عبارت دیگر 3VL و UNK ها، پایه و بنیان تمام مدل‌های رابطه‌ای را به هم می‌زنند.

۱۹,٤. نال‌ها و کلیدها تهی S AND KEYS

با وجود پیام قسمت قبل، حقیقت این است که تهی و 3VL در حین مکاتبه در بسیاری از تولیدات پشتیبانی شده‌اند. علاوه بر این چنین حمایت‌هایی معانی مهمی مخصوصاً برای کلیدها دارد. در این قسمت ما به بررسی چنین مضمون‌هایی خواهیم پرداخت.

کلید اصلی. [1]

همان‌طور که در بخش ۹,۱۰ توضیح داده شد. مدل رابطه‌ای باید ثبت شوند.

کلید اصلی یکی از کلیدهای ثانویه [2]رابطه که طراح انتخاب می‌کند و به سیستم معرفی می‌شود تعریف می‌شود. توجه داشته باشید که از نظر تئوریک کلیدهای ثانویه اعتبار یکسان دارند و هیچ کلید ثانویه‌ای بر دیگری برتری ندارد.

هر رابطه‌ای حداقل یک کلید ثانویه دارد بنابراین هر رابطه‌ای باید کلید اصلی داشته باشد. هر کلید ثانویه غیر از کلید اصلی، کلید دیگری دارد به نام کلید بدیل یا کلید دیگر.

مدل به طور تاریخی شامل محدودیت‌ها و قاعده‌ی جامعیت موجودتی [1] می‌شود:

---

[1] Primary key
[2] Alternate keys

قاعده‌ی جامعیت موجودتی ناظر به کلید اصلی ست و این‌گونه تعریف می‌شود که: هیچ جزئی تشکیل دهنده کلید اصلی نمی‌تواند هیچ مقدار داشته باشد و به عبارتی نمی‌تواند آن را قبول کند.

مبنای چنین قاعده‌ای بر موارد زیر است:

- ✓ تاپل‌ها رابطه‌ای، موجودیت‌ها را در دنیای واقعی نشان می‌دهند.
- ✓ موجودیت در دنیای حقیقی با معلومات قابل تشخیص است.
- ✓ بنابراین قرین‌های آن‌ها در پایگاه داده باید قابل تشخیص باشند.
- ✓ اعتبار کلید اصلی ایمن بماند و مقدارش در پایگاه داده تغییر نکند یا حداقل تغییرات را داشته باشد.
- ✓ بنابراین اعتبار کلیدهای اصلی نباید «نهست» شوند.

۱- قبل از هر چیز، اغلب این‌گونه برآورد می‌شود که قاعده‌ی جامعیت موجودتی بر این اعتقاد است که ارزش‌های کلید اصلی باید منحصر به فرد باشند. اما این طور نیست. درست است که باید چنین مقدار منحصر به فردی به اعتبار کلید اصلی اختصاص یابد اما الزامات با تعریفات کلید اصلی و به جز خود آن درک می‌شوند.

۲- در قدم دوم توجه داشته باشید که تقاضای قاعده صراحتاً بر کلید اصلی باشد: با چنین تلویحی، کلیدهای بدیل اجازه پذیرش هیچ مقدار را خواهد داد. با این وضعیت، کلید بدیل به حکم قاعده‌ی جامعیت موجودتی نباید به عنوان کلید اصلی انتخاب شود. بنابراین در چه حالتی

---

[1] Entity integrity

کلید بدیل در مکان اول یک کلید ثانویه است؟ چنانکه ما باید بگوییم که کلیدهای بدیل نمی‌توانند هیچ مقدارها را بپذیرند، قاعده جامعیت موجودتی نیز اجازه‌ی اصلی شدن را به کلیدهای بدیل نمی‌دهد. در این حالت چنین به نظر می سد که در این قاعده آیتم‌های مبهمی وجود دارد.

۳- و در آخر، توجه داشته باشید که قاعده جامعیت موجودتی تنها رلوار های اساسی را بگیرند چرا که دیگر رلوار ها کلید اصلی‌ای دارند که هیچ مقدار را می‌پذیرند.

برای درک بهتر این موضوع به مثال زیر توجه کنید:

پرتوافکنی رلوار R را روی صفت A ای در نظر بگیرید که هیچ مقدار را نیز می‌پذیرد. قاعده بدین گونه اصول قابلیت تبادل را زیر پا می‌گذارد.

فرض کنید ما دیگر هیچ مقدارها را نمی‌پذیریم و به جای اطلاعات نهست از ارزش‌های ویژه استفاده می‌کنیم که این مقدارهای ویژه گاهی به عنوان مقدارهای پیش فرض و قراردادی شناخته می‌شوند. درست مثل آنچه در دنیای واقعی انجام می‌دهیم. در آن زمان ما ممکن است بخواهیم که نسخه‌ی تعریف شده‌ی قاعده‌ی جامعیت موجودتی را ابقا کنیم. هیچ مؤلفه‌ای از کلید اصلی base relvar اجازه‌ی پذیرش در چنین مقادیر ویژه‌ای را ندارند. در تصویر ۱۹,۲ مثالی در مورد base relvar ای به نام SURVEY آورده شده است. این جدول میانگین، حداکثر و حداقل حقوق افراد متولد در سال y، BIRTHYEAR، را نشان می‌دهد که در اینجا BIRTHYEAR همان کلید اصلی ست؛ و تاپلی که «؟» نشان داده شده، افرادی را نشان می‌دهد که از پاسخ به سوال ما، «در چه سالی متولد شدید؟» خودداری کردند.

SURVEY

BIRTHEAR	AVGSAL	MAXSAL	MINSAL
1960	85K	130K	33K
1961	82K	125K	32K
1962	77K	99K	32K
1963	78K	97K	35K
….	…	…	…
1970	29K	35K	12K
????	56K	117K	20K

کلیدهای خارجی.

پایگاه داده‌ای DEPT-EMP تصویر ۱۹,۱ را دوباره ملاحظه فرمایید. ممکن است دفعه‌ی قبل به این موضوع توجه نکرده باشید اما با کمی تأمل چنین چیزی نگفتیم که #DEPT از رلوار EMP در شکل، یک کلید خارجی بود. اما اکنون فرض کنید که هست. بدیهی ست که قاعده جامعیت ارجاعی به چند فیل‌تر نیاز دارد زیرا کلیدهای خارجی مجبورند تا هیچ مقدارها را بپذیرند؛ و این مقادیر پوچ، قاعده‌ی مذکور را زیر پا می‌گذارند.

- نسخه‌ی اصلی قاعده‌ی جامعیت ارجاعی:

پایگاه داده‌ها نباید حاوی مقداری برای کلید خارجی باشد که در رابطه‌ی مرجع وجود نداشته باشد.

تا زمانی که تعمیم تعریف اعتبار کلید خارجی بی همتا را داریم تعریفی که از این قاعده داده شد را در نظر داشته باشید.

۱- روش هیچ مقدار گذاری در قاعده‌ی ارجاعی: در این روش با حذف تاپل مرجع کلید خارجی در تاپل رجوع کننده هیچ مقدار گذاری می‌شود به شرط آنکه در رابطه‌ی رجوع کننده کلید اصلی رابطه نباشد.

۲- افزایش احتمال پذیرش هیچ مقدار در کلیدهای خارجی احتمال دیگر اعمال ارجاعی را بالا می‌برد. در عمل به هنگام سازی مقدار کلید اصلی در تاپل مرجع، به روش‌هایی در عمل حذف می‌شوند. در این موقعیت در واقع با اجرای دستور به هنگام سازی اول، دستور دوم به هنگام سازی نیز باید به اجرا درآید.

```
VAR SP BASE RELATION {…} …
FOREIGN KEY {S#} REFERENCES S
 ON DELETE SET NULL
ON UPDATE SET NULL
```

به هر حال پایگاه داده‌ها باید در شمای پایگاه نظر خود را در عمل به هنگام سازی و عمل حذف با انتخاب گزینه مناسب به سیستم اعمال کند.

SET تهی تنها برای یک کلید خارجی که در مکان اول اجازه‌ی قرار گیری هیچ مقدار را می‌دهد می‌تواند شناخته شود.

۳- آخرین و مهم‌ترین موضوع این است که این طور به نظر می‌رسد که از احتیاجات مبنی بر پذیرش هیچ مقدار در کلیدهای خارجی می‌توان با طراحی پایگاه داده‌ی مخصوصی پرهیز کرد. مثال DEPT-EMP را دوباره مشاهده کنید. اگر واقعاً برای عدد DEPT این امکان وجود داشته باشد که برای EMP مشخصی، بی‌مقدار توصیف شود در آن صورت واضح است که بهترین راه برای employee این باشد که در رلوار EMP، صفت #DEPT مشمول نشود. اما ترجیحاً با داشتن رلوار جداگانه‌ی ED یا همان وابستگی برابری، و با صفت #EMP و #DEPT، EMPLOYEE تعیین شده در DEPARTMENT تعیین شده قرار گیرد.

۱۹,۵. فرا پیوند. [1]

در این قسمت به اختصار در مورد عمل گر های فرا پیوند بحث می‌کنیم. عمل گر فرا پیوند گونه دیگری از عمل گر پیوند است و تعمیم یافته‌ی پیوندهای معمولی و پیوندهای داخلی ست. در این نوع پیوند علاوه بر تاپل‌های پیوند شدنی از دو رابطه، تاپل‌های پیوند نشدنی از چپ و راست گسترش یافته با هیچ مقدار، در رابطه‌ی جواب وارد می‌شوند.

```
(S JOIN SP)
UNION
(EXTEND ((S{S#} MUNUS SP {S#}) JOIN S)
ADD (NULL AS P#, NULL AS QTY))
```

نتیجه کار تاپل‌هایی برای تأمین کنندگانی ست که هیچ قطعه‌ای را تأمین نمی‌کنند همچنین تاپل‌هایی که با هیچ مقدارها در #P و QTY تعمیم و گسترش یافته‌اند.

اجازه دهید تا این مثال را بیشتر مورد بررسی قرار دهیم. همان‌طور که در شمای تصویر شکل ۱۹,۳ مشاهده می‌کنید قسمت بالا، ارزش و اعتبار برخی از نمونه‌های داده‌ها را برای رلوار های S و SP نشان می‌دهد. قسمت میانی نشان دهنده‌ی پیوند داخلی و معمولی ست و قسمت پایینی نیز چیزی شبیه فرا پیوند به نمایش گذاشته شده است.

همان‌طور که در شکل نیز می‌بینید، پیوند داخلی " LOSES INFORMATION " برای تأمین کننده‌ی S_5 که هیچ قطعه‌ای را تأمین نمی‌کند نشان داده شده است. در حال یکه در فرا پیوند " PRESERVE " و نگهداری چنین اطلاعاتی یکی از مهم‌ترین مسائل فرا پیوند به حساب می‌آید.

مشکل اینجاست که فرا پیوند قصد دارد تا این مشکل را برطرف سازد.

برخی از نویسندگان بر این باورند که به منظور دست‌یابی به نتایج دلخواه بایستی حمایتی مستقیم برای فرا پیوند مهیا شود تا مسیری غیر مستقیم. به هر حال ما خودمان هم برای دلایل زیر مصداق قطعی نداریم و الزاماً آن را تصدیق نمی‌کنیم.

---

[1] Outer join

S

S#	SNAME	STATIS	CITY
S2	JONES	10	Paris
S5	ADAMS	10	Athens

SP

S#	P#	QTY
S2	P1	300
S2	P2	400

Regular (inner) join:

S#	SNAME	STATUS	CITY	P#	QTY
S2	JONES	10	Paris	P1	300
S2	JONES	10	Paris	P2	400

Quter join:

S#	SNAME	STATUS	CITY	P#	QTY
S2	Jones	10	Paris	P1	300
S2	Jones	10	Paris	P2	400
S5	Adams	30	Athens	UNK	UNK

شکل ۱۹,۳

صفت‌های مقدار رابطه‌ای خطمشی دیگری را به را مشکلات موجود بر سر راه ارائه می‌دهند. این خطمشی هیچ مقدارها و همچنین فرا پیوندها را درگیر نمی‌کنند و در این صورت پاسخ‌های ظریفی که به پاسخ‌های رابطه‌ای معروفند را در میان می‌گذارد.

مقدارهای داده‌ی نمونه از قسمت بالای شکل ۱۹,۳ . برای مثال عبارت:

WITH (S RENAME S# AS X) AS Y:
(EXTEND Y ADD (SP WHERE S# = X) AS PQ) RENAME X AS S#

نتیجه‌ای برابر تصویر شماره‌ی ۱۹,٤ را به دنبال خواهد داشت.

S2	Jones	10	Paris	P# QTY: P1 300, P2 400
S5	Adams	30	Athens	P# QTY:

منحصراً به شکل بالا توجه کنید و به همان مقدار تأمین نشده ای که از سوی S5 روی جدول مشخص شده است. همان‌طور که مشاهده می‌کنید تأمین کننده‌ی S5 نتوانسته هیچ قطعه‌ای را تولید و تأمین نماید به همین خاطر در جدول یک قسمت خالی برای آن در نظر گرفته شده است نه یک هیچ مقدار. به عبارت دیگر، چنانچه صفات مقدار رابطه‌ای به خوبی پشتیبانی شده باشند دیگر نیازی به فرا پیوند در این موقعیت نمی‌باشد.

این قسمت را با ملاحظه به امکان تعریف پیشوند «فرا» در بعضی از عمل گر های جبر رابطه‌ای به پایان می‌بریم. خصوصاً در بحث‌های چون اجتماع و به طبع آن فرا اجتماع، توابع و فرا توابع منطقی، و دیگر عمل گر های مختلف.

فرا اجتماع رشته‌ی مشکلات تفسیرها است؛ و نسبت به آن‌هایی که با فرا پیوند درگیر هستند از عملکرد پایین‌تری برخوردارند.

۱۹,٦. مقادیر و ارزش‌های ویژه و تک مقدارها.

همان‌طور که دیدیم هیچ مقدارها مدل رابطه‌ای را به هم می‌زنند. در واقع لازم به توضیح است که مدل رابطه‌ای بیش از ده سال است که بدون آن‌ها مدیریت شده و کار می‌کند. این مدل اولین بار در سال ۱۹٦۹ تعریف شد و تا سال ۱۹۷۹ هیچ خبری از هیچ مقدارها و به عبارتی نال‌ها نبود.

فرض کنید که کل نظریه‌ی هیچ مقدارها را پذیرفته باشیم و از مقادیر ویژه بجای آن‌ها استفاده می‌کنیم تا اطلاعات نهست را به نمایش بگذاریم. توجه داشته باشید که مقادیر ویژه دقیقاً همان چیزی هستند که در دنیای واقعی اتفاق می‌افتند. در دنیای حقیقی، برای مثال، ممکن است که ما مقدار ویژه‌ی ”?“ را برای تشخیص و تفکیک ساعت‌هایی که برای کار یک employee ست در نظر گرفته می‌شود. در این صورت

چیزی که ما از این موضوع متوجه می‌شویم این است که مقدار اصلی به دلایلی برای ما ناشناخته است. در نتیجه ما بجای اینکه از مقادیر معمول برای پاسخ به این سوال استفاده کنیم از مقدار ویژه استفاده کرده‌ایم. البته توجه داشته باشید که نوع مقدار ویژه باید از نوع کاربردی آن مقدار باشد: مثلاً در مثل ساعت‌های کار، صفت ساعت‌های کارکرد یا همان HOURS_WORKED لزوماً کلمه‌ای برای ذخیره‌ی چنین مثالی نیست.

۱۹,۷. ساختارهای SQL

SQL مخفف عبارت Structured Query Language (زبان جستجویی ساخت یافته) است و به کاربر امکان اتصال و دسترسی به اطلاعات موجود در یک پایگاه داده را می‌دهد.

– زبان SQL قادر است تا برای یک پایگاه داده عمل جستجو و گزینش اطلاعات را انجام دهد و همچنین اطلاعات ذخیره شده در یک پایگاه داده را بازیابی، حذف، ذخیره، اضافه و یا به روز کند.

– زبان SQL یک استاندارد بین‌المللی است.

SQL بجای دو اصطلاح رابطه و متغیر رابطه‌ای از اصطلاح جدول استفاده می‌کند.

تا تبدیل شدن به یک زبان رابطه‌ای کامل فاصله زیادی دارد. با این همه SQL استاندارد است و تقریباً توسط هر محصول نرم افزاری بانک اطلاعات پشتیبانی می‌شود و هر فرد حرفه‌ای نیازمند آن است.

SQL در واقع از هیچ مقدارها حمایت می‌کند و چنین حمایتی در این زبان کار به سایر پیچیده‌ای به شمار می‌رود. اگر چه ما تنها می‌گوییم که SQL، منطق سه ارزشی را دنبال می‌کند حقیقت این است که در این حمایت و دنباله روی مشکلات بسیار زیادی برای آن منطق به وجود می‌آید.

انواع داده.

همان‌طور که در بخش ٤ هم دیدیم شامل انواع BOOLEAN که در سال ۱۹۹۹ جزو استانداردها معرفی شد. عمل گر های متداول بولین نیز عبارتند از : AND, OR, NOT . عبارات بولینی نیز هر جا که عبارت‌های اسکالر باشند ظهور پیدا خواهد کرد. الآن می‌دانیم که تنها دو مقدار حقیقی TRUE و FALSE وجود ندارد بلکه علاوه بر این دو مقدار، مقدار دیگری نیز که قبلاً هم در مورد آن بسیار صحبت کردیم وجود دارد که به آن UNK گویند همانند مقدار ناشناخته‌ای که صفت‌های ناشناخته را شامل می‌شود و اگر چیزی چنین ارزش و مقداری را پیدا می‌کرد به صراحت می‌توان گفت که هیچ چیزی نبوده است.

ارزش UNKNOWN برای متغیر B از نوع BOOLEAN در واقع همان مقدار هیچ مقدار و نال می‌باشد.

بعد از چنین معادلی، در مقام مقایسه B=UNKNOWN پاسخ دیگر TRUE نخواهد بود و جواب تهی را خواهد داد.

صرف‌نظر از مقدار B، چنین مقایسه‌ای برای B=UNKNOWN همواره تهی بوده زیرا منطقاً چنین مقداری برای هیچ مقدار است.

جداول پایه.

در اساس همان است که در بخش ٦ قسمت ٦,٦ توضیح دادیم. اما با برخی اصطلاحات دیگر مانند اعلان یکتایی مقادیر ستون و معرفی کلید اصلی، معرفی کلیدهای خارجی و گزینه‌های مشخص کردن روش اعمال قاعده‌ی جامعیت ارجاعی.

اگر کلید خارجی یک ستون ساده باشد می‌توان در همان کلاس معرفی ستون نیز جدول مرجع را معرفی کرد.

چنانچه بخواهیم در چنین جداولی در قسمت ستون‌های آن ارزش TRUE را داشته باشیم باید از هیچ مقدارها صرف‌نظر کنیم و در مورد آن‌ها در مشخصات کلید اصلی صحبتی به میان نبریم.

جدول عبارتی.

در قسمت ۸,٦ نیز به این موضع پرداخته شد؛ عمل گر پیوند طبیعی گونه‌ای از عمل گر پیوند است علاوه بر اینکه در مجموعه عنوان رابطه جواب، نام صفت یا صفات پیوند یک‌بار قید می‌شود. از انواع دیگر آن فرا پیوندهای چپ، فرا پیوند راست و فرا پیوند کامل است. در حالت فرا پیوند چپ علاوه بر تاپل‌های پیوند شدنی از دو رابطه، تاپل‌های پیوند نشدنی از رابطه‌ی چپ هم گسترش یافته با هیچ مقدار در رابطه‌ی جواب رد می‌شود و همین منوال برای فرا پیوند از راست تکرار می‌شود. فرا پیوند کامل نیز هر دو حالت راست و چپ را شامل می‌شود.

```
S LEFT JOIN SP ON S.S# = SP.S#
SLEFT JOIN SP USING (S#)
S LEFT NATURAL JOIN SP
```

این سه عبارت هر سه برابر هم می‌باشند و به عبارتی معادل یکدیگر هستند اما تنها تفاوتی که در آن‌ها وجود دارد تعداد ستون‌های جداول مربوط به آن‌هاست؛ به طوری که در اولین عبارت، دو ستون برابر که هر دو S# هستند تولید می‌شود و در دومین و سومین عبارت تنها یک ستون دارا می‌باشند.

SQL تقریباً حمایتی کاملی نیز از فرا اجتماع می‌نماید که به پیوند اجتماع گویند و در سال ۱۹۹۲ در SQL پدید آمد و در سال ۲۰۰۳ نیز حذف شد.

عبارات بولینی BOOLEAN Expression

عبارات بولینی SQL، در هر موقعیتی تحت تأثیر هیچ مقدارها و منطق سه ارزشی قرار می‌گیرند. ما در این قسمت به بررسی چند مورد مهم می‌پردازیم:

- تست‌هایی برای هیچ مقدار: SQL دو عمل گر ویژه‌ی مقایسه‌ای را ارائه می‌کند؛ یکی تهی و دیگری IS NOT تهی است که برای تشخیص حضور و غیاب هیچ مقدار به کار می‌روند. از لحاظ برنامه نویسی عبارت است از :

<row value constructor> IS [NOT] تهی

چنانچه <row value constructor> یک ردیف یک درجه‌ای را بسازد در آن صورت SQL با آن طوری رفتار می‌کند انگار که آن واقعاً مقدار مشمول در آن سطر را تفکیک می‌کند.
به عنوان موضوع آخر، به ROW در جدول به شکل‌های مختلفی نگاه می‌شود که از جمله‌ی آن‌ها عبارتند از:

۱- Row یک تهی و هیچ مقدار است؛ اگر و تنها اگر هر مؤلفه‌ای در آن هیچ مقدار باشد.
۲- Row یک non تهی و غیر هیچ مقدار است؛ اگر و تنها اگر هر مؤلفه‌ای در آن non تهی باشد.

یکی از پیامدهای این مشکل آن است که اگر r یک سطر با دو مؤلفه باشد که c1 و c2 خوانده شوند؛

سپس عبارت r IS NOT تهی و NOT (r IS) تهی با هم برابر خواهند بود.
اولین برابری این‌گونه است: c1 IS NOT تهی AND c2 IS NOT تهی.

دومین برابری نیز به این صورت خواهد بود: c1 IS NOT تهی OR c2 IS NOT تهی .نتیجه‌ی دیگر این است که اگر r شامل چند مؤلفه‌ی تهی و non تهی باشد در آن صورت r خودش ظاهراً نه تهی است نه non تهی .

- تست‌هایی برای true, false, UNKnown؛ اگر p یک عبارات بولین در پرانتز باشد. (در واقع این پرانتزها بعضی اوقات ضروری نیستند اما هیچ‌گاه اشتباه نیز نیستند). سپس عبارت زیر همان عبارت بولین خواهد بود:

P IS (NOT) TRUE
P IS (NOT) FALSE
P IS (NOT) UNKNOWN

معنی این عبارت در جدول زیر آمده است لطفاً جدول مورد نظر را ملاحظه فرمایید:

P	True	false	unk
P IS TRUE	TRUE	FALSE	FALSE
P IS NOT TREU	FALSE	TRUE	TRUE
P IS FALSE	FALSE	TRUE	FALSE
P IS NOT FALSE	TRUE	FALSE	TRUE
P IS UNKNOWN	FALSE	FALSE	TRUE
P IS NOT UNKNOWN	TRUE	TRUE	FALSE

همان‌طور که مشاهده می‌کنید، عبارت p IS NOT TRUE و NOT p با هم برابر نیستند.

- شرایط EXIST: عمل گر EXIST در SQL همان سور وجودی در منطق سه ارزشی نیست زیرا حاصل ارزیابی این عبارات تنها ممکن است

TRUEیا FALSE شود نه UNK حتی زمانی که ارزش UNK از لحاظ منطقی درست باشد. FALSE ارزیابی زمانی ست که در جدول قسمت مورد نظر خالی باشد و TRUE بر عکس همین حالت است.

- شرایط UNIQUE: امکانی ست برای وارسی تمایز سطرهای جدول اگر در جدول جواب تمام سطرها از یکدیگر متمایز باشند نتیجه‌ی UNIQUE، TRUE خواهد بود و در غیر این صورت FALSE می‌باشد. شرایط UNIQUE هم تقریباً شرایطی برابر با EXIST دارد به این شکل که اگر جدول جواب تنها یک سطر داشته باشد و یا تهی باشد به جای اینکه ارزش UNK را ارزیابی کنند چه بسا از لحاظ منطقی هم درست باشد، پیام true را خواهد داد.

- در SQL ممکن است نتیجه پرس و جو حاوی داده‌های تکراری باشد لذا برای حذف مقادیر تکراری کلمه distinct را بایستی بعد از select بکار برد. در شرایط DISTINCT چنانچه دو سطر شبیه هم باشند باید سطرها را به دو قسمت چپ و راستی و LEFT AND RIGHT تقسیم بندی کرد مؤلفه‌های هر کدام را به ترتیب چنین تعریف می‌کنیم: Li و Ri به شرطی که i = 1, 2, 3, …, n و البته باید هر دو از یک درجه باشند. وارسی وضعیت قیاسی Li و Ri باید Li = Ri را به ما بدهد در آن صورت عبارت :

Left IS DISTINCT FROM Right

برابر false خواهد بود اگر به ازای هر iای داشته باشیم الف- Li = Riب- Li و Ri هر دو هیچ مقدار هستند. در غیر این صورت true و درست خواهد بود.

- Literals: گاهی اوقات ممکن است که نوعی از literal نماینده‌ی نمایش هیچ مقدار باشد مثلاً در جمله‌ی INSERT .نه البته در همه‌ی موارد. تنها از زمانی

می‌توان از LITERAL استفاده کرد که خود آن اجازه‌ی پذیرش آن را داشته باشد مثلاً زمانی که می‌توان از تهی برای تفکیک استفاده کرد مانند WHERE X = تهی .

- COALESCE: آنالوگ SQL عمل گر IF_UNK است. به بیان دقیق‌تر و تأمل بیشتر متوجه می‌شویم که عبارت COALESCE (x, y, … , z) به مانند نال و هیچ مقدار می‌ماند اگر x, y, … , z به طور کلی نال و هیچ مقدار ارزیابی شوند.

- عمل گر های جمعی و گروهی (Aggregate operators) : عمل گر های جمعی و گروهی SQL مانند SUM, AVG,… بر اساس قواعد عمل گر های اسکالر توضیح داده شده در قسمت ۱,۲ رفتار نمی‌کنند. اما در عوض از تمام هیچ مقدارها صرف‌نظر می‌کنند البته به جز در عمل گر COUNT که در آن هیچ مقدارها ماند مقادیر معمولی رفتار می‌کنند. همچنین اگر مواقعی که در چنین اپراتورهایی قسمتی خالی ارزیابی شود در آن صورت COUNT صفر را به عنوان ارزش ارزیابی می‌کند.

- Scalar sub queries (پرسش‌های فرعی یا ریز پرسش‌های عددی): چنانچه عبارت عددی یک عبارت جدولی در داخل پرانتز باشد مثل.

(SELECT S.CITY FROM S WHERE S.S# = S#) ‘S1

سپس عبارت جدولی برای ارزیابی سطر و ستون نهایی مورد نیاز خواهد بود.
چنانچه عبارت مذکور یک جدول با یک ستون و بدون سطر را ارزیابی کند در آن صورت SQL مقدار عبارت عددی را تهی و هیچ مقدار تعریف می‌کند.

کلیدها.

کلیدهای ثانویه؛

فرض کنید C ستون مؤلفه‌ی کلید ثانویه K در جدول اساسی باشد. اگر K کلید اصلی باشد، مقدار C نیز هیچ‌گاه هیچ مقدار نخواهد بود. اگر کلید اصلی نباشد، SQL این اجازه را می‌دهد تا C هر تعداد از هیچ مقدار که بخواهد به خود بگیرد.

فرض کنید K_2 مقدار جدیدی از K باشد که برخی آن را با INSERT یا UPDATE معرفی کرده‌اند. این دو عمل گر مؤلفه جدید را مرجوع خواهند کرد اگر مقدار این مؤلفه‌ی جدید دقیقاً برابر مؤلفه K_1 باشد. اینکه دو مؤلفه‌ی K_1 و K_2 مثل هم باشند چه معنایی می‌تواند بدهد؟

بدین معنا خواهد بود که هیچ دو رابطه‌ای از سه جمله‌ی زیر با هم برابر نیستند:

۱- K_1 و K_2 به منظور مقایسه با هم برابرند.
۲- K_1 و K_2 به منظور یگانگی کلید ثانویه با هم برابرند.
۳- K_1 و K_2 به منظور حذف عبارات دو تایی با هم برابرند.

جمله اولی بر اساس قواعد مربوط به منطق سه ارزشی ست.

جمله‌ی دوم بر اساس قواعد یکتایی و UNIQUE تعریف شده است.

و جمله‌ی سوم بر اساس تعریف عبارات المثنی و دوبل در قسمت ۱۹,۲ در میان گذارده شده است.

کلیدهای خارجی؛

با توجه به مفهوم میدان و دامنه‌ی ثانویه، کلید خارجی رابطه‌ای ست که از میدان ثانویه یا میدان اصلی مقدار می‌گیرد و می‌تواند مانند کلید ثانویه باشد. در این

صورت صفت کلید خارجی نیز باید با صفت کلید ثانویه هم میدان باشد. از بحث در مورد جزئیات صرف‌نظر می‌کنیم.

توجه: هیچ مقدار عملی منطقی برای اعمال ارجاعی مانند CASCADE, SETتهی می‌باشد مخصوصاً در ON UPDATE و ON DELET کردن.

SQL ادغام شده:

ادغام احکام SQL در یک زبان میزبان در هر نسخه استاندارد این زبان انجام شدنی ست. در حال حاضر این زبان را می‌توان در اکثر زبان‌های برنامه سازی رایج ادغام کرد. در SQL اصل دو اسلوبی بودن زبان رعایت شده است. یعنی هر حکم این زبان که بتوان آن را به صورت تعاملی به کار برد را می‌توان در یک برنامه‌ی کاربردی نوشته شده به یک زبان میزبان نیز استفاده کرد.

متغیر ثانویه:

مثال زیر در فصل ٤ نیز تکرار شده بود، لطفاً به این مثال توجه کنید که مثالی از SQL ادغام شده با SELECT یگانه است:

```
EXEC SQL SELECT STATUS, CITY
    INTO: RANK,  : TOWN
    FROM S
    WHERE S# = S# (: GIVENS#);
```

فرض کنید که STATUS می‌تواند برای بعضی از تأمین کنندگان، هیچ مقدار باشد. سپس جمله‌ی SELECT اگر STATUS هیچ مقدار باشد، FAIL را نمایش می‌دهد.

به طور کلی چنانچه امکان هیچ مقدار شدن برای برخی از آیتم‌های SELECT وجود داشته باشد، کاربر باید مهارت خاصی در متغیرهای ثانویه داشته باشد.

```
EXEC SQL SELECT STATUS, CITY
      INTO: RANK INDICATOR: RANKIND,  : TOWN
```

```
FROM S
WHERE S# = S# (: GINENS#);
IF RANKIND = -1 THEN/ * STATUS WAS NULL*/ …; END IF;
```

اگر آیتم بازیابی شده، هیچ مقدار باشد و متغیر ثانویه هم مشخص شده باشد در آن صورت آن متغیر مقدار 1- را خواهد داشت.

- نظم و آراستگی (Ordering):

ORDER BY به منظور آراستن نتایج سطور حاصل از ارزیابی عبارات جدولی استفاده می‌شوند البته ممکن است برای سوالات تعاملی (اندر کنشی) نیز به کار روند.

سوال اینجاست که نظم‌های مذکور برای دو مقدار عددی A و B چگونه است اگر A یا B یا هر دوی آن‌ها هیچ مقدار باشند؟

پاسخ SQL این‌گونه خواهد بود که :

۱- در این زمینه، تمام هیچ مقدارها تماماً شبیه به یکدیگر و برابر یکدیگرند.

۲- در این زمینه، هیچ مقدار یا بزرگت ریا کوچک‌تر از مقادیر Non تهی به شمار می‌روند.

۱۹،۸. خلاصه.

در این گفتار در مورد مشکلات اطلاعات نهست با توجه بهمشی مشکلات موجود بر سر راه هیچ مقدارها و منطق سه ارزشی صحبت کردیم؛ و بر این نکته تاکید داشتیم که هیچ مقدار یا همان تهی، خود به عنوان یک مقدار به حساب نمی‌آید. در منطق سه ارزشی صفاتی که هیچ مقداری برای آن‌ها مشخص نبود و در حقیقت هیچ چیزی نبودند که به اصطلاح ما هیچ مقدار بودند در این منطق به عنوان ناشناخته‌ها یا UNKnown معرفی شدند. همچنین در مورد انواع هیچ مقدارها هم بحث کردیم و

UNK را معرفی کردیم که مختصر عبارت مقادیر ناشناخته یا همان value-UNKnown kind بود.

سپس در مورد دلالت UNK ها و منطق سه ارزشی برای عمل گر های بولینی صحبت کردیم که از جمله‌ی این عمل گر ها عبارتاند از : AND, OR, NOT .

در مورد به هنگام سازی عمل گر های INSERT و UPDATE صحبت شد و در این حین اپراتورهای IS_UNK که برای تست UNK و IF_UNK که برای تبدیل UNK به مقدار غیر UNK ای استفاده می‌شدند، معرفی شدند. در این مورد بحث کردیم که UNK همان UNK نیست و از لحاظ مقداری آن‌ها گفت و گو کردیم و گفتیم که کدام مقدار یا همان ارزش را دارند و کدام ندارند.

در مورد عبارات برابر یا معادل در منطق سه ارزشی بحث کردیم و گفتیم که در این منطق چه موقع باید دنیای حقیقی را در نظر گرفت و چه موقع نباید.

سپس در مورد دلالت هیچ مقدارها بر کلیدهای اصلی و خارجی و قاعده‌ی جامعیت موجودتی و قاعده‌ی جامعیت ارجاعی توضیح دادیم. مطالبی نیز در مورد فرا پیوندها اعم از چپ و راست و کامل و فرا اجتماع گفتیم.

مختصری در مورد رفتار SQL بر اطلاعات نهست بر اساس منطق سه ارزشی گفته شد.

این قسمت را به چند گفته‌ی دیگر به پایان می‌بریم:

- ما این موضوع را روشن کردیم که اگر شما نسبت به مشکلات 3VL قانع نشدید، به شما توصیه می‌کنیم تا از پیامدهای SQL به پرهیزید زیرا این عیب همیشه مورد بحث و گفت و گو بوده است.
- توصیه‌ی ما به کاربران DBMS این است که از حمایت 3VL ای فروشندگان چشم پوشی کنند و از شمای مقادیر ویژه‌ی مرتب شده به جای شماهایی که در مرجع ۱۹,۱۲ گفته شده استفاده کنند.

- و در نهایت بار دیگر نکته‌ی اساسی‌ای را که در قسمت ۱۹,۳ برای شما بیان کردیم را تکرار می‌کنیم. به آرامی بخوانید! – اگر مقدار صفت در تاپل داده شده و در رابطه‌ی داده شده، هیچ مقدار باشد در آن صورت موقعیت صفت در واقع هیچ چیز خواهد بود. در این صورت صفت، صفت نخواهد بود و همچنین تاپل و رابطه‌ی، خود تاپل و رابطه نخواهند بود؛ و در نهایت دیگر یک تئوری رابطه‌ای ریاضیاتی نخواهد بود.