I

A relációs adatbázisok és az SQL

1

Mit jelent az, hogy „relációs”?

„A tudás egy kis része annak a tudatlanságnak,
amit rendszerezünk és osztályozunk.”
– Ambrose Bierce

A fejezet témakörei

- Az adatbázisok fajtái
- A relációs modell rövid története
- A relációs adatbázisok felépítése
- Miért hasznosak számunkra a relációs adatbázisok?
- Összefoglalás

Mielőtt fejest ugranánk az SQL világába, nem árt, ha rendelkezünk némi háttérinformációval a relációs adatbázisokról, ezért először arról fogunk beszélni, hogy miért találták ki a relációs adatbázisokat, hogyan épülnek fel, és miért hasznosak a számunkra. Ez az az alap, amelyen biztosan kell állnunk, hogy megérthessük az SQL lényegét, és tisztán lássuk, hogy miként aknázhatjuk ki legjobban az SQL képességeit.

Az adatbázisok fajtái

Mit nevezünk adatbázisnak? Amint azt bizonyára tudjuk, az adatbázis adatok rendezett gyűjteménye, amellyel valamilyen szervezett dolgot vagy szervezési folyamatot modellezünk. Tulajdonképpen nem számít, hogy papírt vagy egy számítógépes programot használunk az adatok összegyűjtésére és tárolására: amíg az adatokat konkrét célra, valamilyen rendezett formában gyűjtjük és tároljuk, adatbázisról beszélhetünk. A továbbiakban persze azt fogjuk feltételezni, hogy az adatgyűjtésre és az adatok kezelésére számítógépprogramot használunk.

Az adatbázis-kezelés területén általánosságban kétféle adatbázis van használatban: *műveleti adatbázisok* és *elemző adatbázisok*. Ma szerte a világon műveleti adatbázisok képezik számos vállalat, szervezet és intézmény gerincét. Ezt a fajta adatbázist elsősorban mindennapi adatgyűjtésre, -módosításra és -kezelésre használják. A tárolt adatok *dinami-*

kus adatok, ami azt jelenti, hogy folyamatosan változnak, és mindig friss információkat tükröznek. Az olyan szervezetek, mint az áruházak, a gyártócégek, a kórházak és klinikák, valamint a könyvkiadók műveleti adatbázisokat használnak, mivel az adataik percről percre változnak.

Ezzel szemben az elemző adatbázisok korábbról származó, adott időponthoz kapcsolódó adatokat tárolnak és rendszereznek. Az elemző adatbázisok a változások irányának nyomon követésében hasznosak, valamint a hosszú idő alatt összegyűjtött statisztikai adatok megjelenítését, illetve a taktikai és stratégiai üzleti előrejelzések készítését segítik. Az ilyen adatbázisok *statikus adatokat* tárolnak, vagyis az adatok soha (vagy csak nagyon ritkán) módosulnak, új adatok felvételére viszont gyakran sor kerülhet. Az elemző adatbázisokból kinyert információk általában nem naprakészek: egy adott időpontban készített pillanatfelvételt mutatnak az adatokról. Ilyen fajta adatbázist használnak például a kémiai laboratóriumok, a földmérő vállalatok, illetve a piackutató és -elemző cégek.

A relációs modell rövid története

Az adatbázismodelleknek többféle típusa létezik. Egyeseket – például a hierarchikus és hálózati modelleket – csak régi rendszereken használnak, míg mások – például a relációs modell – széles körben elterjedtek. Más könyvekben az objektum alapú, objektumrelációs és OLAP (online analytical processing, elektronikus elemző-feldolgozó) modellekkel is találkozhatunk. Az SQL-szabvány bővítményeket határoz meg ezeknek a modelleknek a támogatására, és léteznek kereskedelmi adatbázisrendszerek, amelyek meg is valósítanak egyes bővítményeket ezek közül. Mi azonban szigorúan a relációs modellre és a nemzetközi SQL-szabvány magjára fogunk összpontosítani.

A kezdetek

A relációs adatbázis elve 1969-ben született meg, és azóta valószínűleg a legszélesebb körben használt modellé vált az adatbázis-kezelésben. A relációs modell atyja, Dr. Edgar F. Codd (1923–2003) az IBM kutatójaként dolgozott az 1960-as években, és azt vizsgálta, hogy milyen új módszerekkel lehet nagy adatmennyiségeket kezelni. Az akkor létező adatbázismodellekkel és -termékekkel elégedetlen volt, ami arra indította, hogy matematikai elvek és szerkezetek segítségével próbálja meg megoldani az előtte álló tömérdek feladatot. Matematikusként szilárdan hitt benne, hogy a matematikának léteznek olyan ágai, amelyeket alkalmazva megoldhatók az olyan problémák, mint az adatismétlődés (redundancia) kiküszöbölése, az adatok egységességének biztosítása, illetve annak leküzdése, hogy az adatbázisok szerkezete túlzottan függjön a fizikai megvalósítástól.

Dr. Codd hivatalosan 1970 júniusában, az *A Relational Model of Data for Large Shared Databases* című, mérföldkőnek számító munkájában[1] mutatta be az általa kidolgozott új relációs modellt. A modell a matematika két ágára támaszkodott: a halmazelméletre és

[1] *Communications of the ACM*, 1970. június, 377-87. oldal

az elsőrendű predikátumlogikára. Maga a modell is a halmazelmélet egyik kulcsfogalma, a *reláció* kifejezés után kapta a nevét. (Széles körben elterjedt tévedés, hogy a relációs modell neve onnan ered, hogy a relációs adatbázisok táblái között kapcsolatok állhatnak fenn. Most, hogy tudjuk az igazságot, nyugodtan alhatunk.) Szerencsére nem kell részleteiben ismernünk a halmazelméletet vagy az elsőrendű predikátumlogikát ahhoz, hogy relációs adatbázisokat tervezhessünk és használhassunk. Ha megfelelő adatbázis-tervezési módszert követünk – például a Mike Hernandez *Database Design for Mere Mortals* (Addison-Wesley, 2004; magyarul: *Adatbázis-tervezés: A relációs adatbázisok alapjairól földi halandóknak*. Kiskapu, 2004) című könyvében ismertetett eljárásokat –, egészséges és hatékony adatbázis-szerkezetet alakíthatunk ki, amelyet bizalommal használhatunk mindenféle adat összegyűjtésére és kezelésére. (Nos rendben, természetesen egy kicsit azért *muszáj* értenünk a predikátumokhoz és a halmazelmélethez, ha bonyolultabb feladatokat is meg szeretnénk oldani. A predikátumokkal – állításokkal vagy szűrőkkel; a „predikátum" valójában csak egy fellengzős név – kapcsolatos alapvető tudnivalókat a 6., a halmazelmélet alapjait pedig a 7. fejezetben tárgyaljuk.)

A relációs adatbázisprogramok

Megjelenése óta a relációs modell az alapja az RDBMS (relational database management system, relációs adatbázis-kezelő rendszer) néven ismert adatbázistermékeknek. Ilyen terméket számos gyártó készít, és az évek során a legkülönbözőbb iparágak és szervezetek vették át a használatát, akik sokféle környezetben alkalmazzák. Az 1970-es években a nagyszámítógépek olyan programokat használtak, mint az IBM által kifejlesztett *System R* vagy a Berkeley-i Kaliforniai Egyetemen megalkotott *INGRES*. A nagygépekhez készített RDMBS rendszerek fejlesztése az 1980-as években olyan programokkal folytatódott, mint az Oracle Corporation *Oracle*, illetve az IBM *DB2* rendszere, a személyi számítógépeknek az 1980-as évek közepén bekövetkezett elterjedése pedig olyan programokat szült, mint az Ashton Tate *dBase*, az Ansa Software *Paradox* vagy a Microrim *R:BASE* szoftvere. Amikor az 1980-as évek végén, illetve az 1990-es évek elején nyilvánvalóvá vált, hogy igény mutatkozik a PC-k közötti adatmegosztásra, megszületett az ügyfél–kiszolgáló rendszerek elve, amelyet a központi helyen tárolt, közösen használható, könnyen kezelhető és biztonságossá tehető adatok ötlete kísért. Erre az elvre olyan termékek épültek, mint az Oracle *Oracle 8i*, valamint a Microsoft *SQL Server* rendszere. Körülbelül 1996 óta már abban az irányban is összehangolt erőfeszítéseket tesznek, hogy az adatbázisok elérhetőségét az Interneten is biztosítsák. A szoftvergyártók komolyan veszik a kérdést, és olyan termékekkel igyekeznek válaszolni a kihívásokra, amelyek webközpontúbbak: ilyen az Allaire cég *Cold Fusion*, a Sybase *Sybase Enterprise Application Studio* vagy a Microsoft *Visual Studio* programcsomagja. A webfejlesztésben az egyik legnépszerűbb adatbázis-kezelő a MySQL AB nyílt forrású *MySQL*-je. Ezt eredetileg Linux rendszerű webkiszolgálókhoz tervezték, de már Microsoft Windows rendszereken futó változata is létezik.

A relációs adatbázisok felépítése

A relációs modellnek megfelelően az adatok a relációs adatbázisokban *relációkban* tárolódnak, amelyeket a felhasználó táblák formájában érzékel. Minden reláció *egyedekből* (tuple, rekord) és *jellemzőkből* (mező) épül fel. Ezen kívül a relációs adatbázisoknak természetesen más tulajdonságaik is vannak – ezekkel foglalkozunk az alábbiakban.

Táblák

A táblák az adatbázisok fő szerkezeti elemei. Minden tábla mindig egyetlen konkrét tárgyat ábrázol. A rekordok és mezők logikai sorrendje a táblán belül egyáltalán nem számít. Legalább egy mezőt – az úgynevezett *elsődleges kulcsot*, amely egyedileg azonosítja a tábla egyes rekordjait– minden táblának tartalmaznia kell. (Az 1.1. ábrán például a CustomerID a Customers tábla elsődleges kulcsa.) Valójában a relációs adatbázisokban levő adatok a táblák ez utóbbi két tulajdonságának köszönhetően függetlenek attól, hogy fizikailag miként tárolódnak a számítógépen. Ez a felhasználóknak előnyös, mert így ahhoz, hogy kinyerjenek egy adatot, nem kell ismerniük az adatot tároló rekord fizikai helyét.

Customers

Customers	FirstName	LastName	StreetAddress	City	State	ZipCode
1010	Angel	Kennedy	667 Red River Road	Austin	TX	78710
1011	Alaina	Hallmark	Route 2, Box 203B	Woodinville	WA	98072
1012	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006
1013	Rachel	Patterson	2114 Longview Lane	San Diego	CA	92199
1014	Sam	Abolrous	611 Alpine Drive	Palm Springs	CA	92263
1015	Darren	Gehring	2601 Seaview Lane	Chico	CA	95926

REKORDOK

MEZŐK

1.1. ábra

Példa egy táblára

A tárgy, amelyet egy adott tábla ábrázol, egy *objektum* vagy egy *esemény* lehet. Ha a tárgy egy objektum, a tábla valami olyat ábrázol, ami „megfogható": egy személyt, egy helyet vagy más fizikai dolgot. Objektumokként táblában ábrázolhatók például pilóták, termékek, gépek, hallgatók, épületek, berendezések, és így tovább. A típusától függetlenül minden objektumnak vannak olyan jellemzői, amelyek adatként tárolhatók, és ezeket az adatokat aztán szinte végtelen módon lehet feldolgozni. Erre a fajta táblára az 1.1. ábra mutat jellegzetes példát.

Amennyiben a tábla tárgya egy esemény, a tábla valami olyasmit ábrázol, ami egy adott időpontban történt, illetve történik (majd), és olyan jellemzői vannak, amelyeket rögzíteni szeretnénk. Ezeket a jellemzőket pontosan ugyanúgy tárolhatjuk adatként, és dolgozhatjuk fel mint információt, mint azokban a táblákban, amelyek valamilyen konkrét objektumot

írnak le. Olyan eseményeket rögzíthetünk például, mint a bírósági tárgyalások, az osztalékkifizetés, a laborleletek vagy egy földtani mérés. Az 1.2. ábrán például egy olyan eseményt ábrázoló táblát láthatunk, amelyet életünk során mindnyájan átélünk – egy orvosi vizetet.

Patient Visit

PatientID	VisitDate	VisitTime	Physician	BloodPressure	Temperature
92001	2006-05-01	10:30	Ehrlich	120 / 80	98.8
97002	2006-05-01	13:00	Hallmark	112 / 74	97.5
99014	2006-05-02	9:30	Fournier	120 / 80	98.8
96105	2006-05-02	11:00	Hallmark	160 / 90	99.1
96203	2006-05-02	14:00	Hallmark	110 / 75	99.3
98003	2006-05-02	9:30	Fournier	120 / 82	98.6

1.2. ábra

Eseményt ábrázoló tábla

Mezők

A mező az adatbázis legkisebb szerkezeti egysége, amely a hozzá tartozó tábla tárgyának egy jellemzőjét írja le. A mezők azok a szerkezetek, amelyek az adatokat ténylegesen tárolják. A mezőkben levő adatokat azután kinyerhetjük és információként megjeleníthetjük, szinte bármilyen elképzelhető formában. Véssük az eszünkbe, hogy az adatokból kinyert információk minősége egyenesen arányos az idő mennyiségével, amit arra fordítunk, hogy maguknak a mezőknek az adat- és szerkezeti következetességét biztosítsuk. A mezők fontosságát nem lehet túlbecsülni.

Egy megfelelően megtervezett adatbázisban minden mező kizárólag egyetlen értéket tartalmaz, és a neve azonosítja a benne tárolt érték típusát. Így nagyon egyszerűvé válik az adatok bevitele a mezőkbe: ha olyan nevű mezőket látunk, mint a FirstName (Keresztnév), a LastName (Vezetéknév), a City (Város), a State (Állam/Megye) vagy a ZipCode (Irányítószám), pontosan tudni fogjuk, hogy milyen értéket kell írnunk az egyes mezőkbe, és az adatok rendezése (például állam szerint) vagy az összes olyan személy kikeresése, akinek a vezetékneve Viescas, ugyancsak könnyű lesz.

Rekordok

A rekord egy adott tábla tárgyának egy egyedi példányát jelképezi, amely a tábla adott sorában található összes mezőt tartalmazza, függetlenül attól, hogy az egyes mezőkben szerepelnek-e értékek. A táblák meghatározásának módjából következik, hogy minden rekordot egy egyedi érték azonosít a teljes adatbázisban, és ez az érték a rekord elsődleges kulcs mezőjében tárolódik.

Az 1.1. ábrán például minden rekord egy-egy különböző vásárlót jelöl a táblában, és az adatbázisban az egyes vásárlókat a CustomerID (Vásárlóazonosító) mező azonosítja. Amint említettük, az egyes rekordok a tábla adott sorában található összes mezőt tartal-

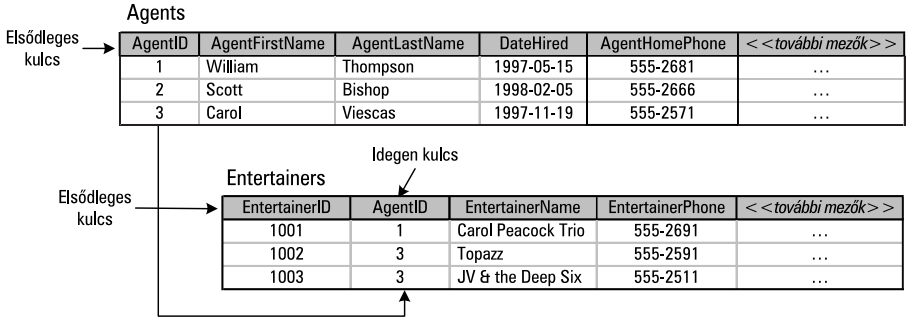
mazzák, és minden mező egy-egy tulajdonságát írja le a rekord által ábrázolt vásárlónak. A rekordok kulcsfontosságúak a táblakapcsolatok megértésében, mivel tudnunk kell, hogy egy tábla rekordjai milyen viszonyban állnak egy másik tábla rekordjaival.

Kulcsok

A kulcsok különleges mezők, amelyek meghatározott szerepet töltenek be az egyes táblákban. Ezt a szerepet a kulcs típusa határozza meg. Bár egy tábla többféle kulcsot tartalmazhat, mi csak a két legfontosabb típust tárgyaljuk: az *elsődleges kulcsot* és az *idegen kulcsot*.

Az elsődleges kulcs olyan mező vagy mezőcsoport, amely egyedileg azonosítja a táblában található összes rekordot. (Ha az elsődleges kulcs két vagy több mezőből áll, *összetett elsődleges kulcsról* beszélünk.) Két oka van annak, hogy miért az elsődleges kulcs a legfontosabb: az *értéke* egy *konkrét rekordot* azonosít a teljes adatbázisban, a *mezője* pedig ugyanott egy *adott táblát*. Az elsődleges kulcsok emellett a tábla szintjén biztosítják a következetességet, és segítenek kapcsolatokat kialakítani más táblákkal. Az adatbázisaink minden táblájának rendelkeznie kell elsődleges kulccsal.

Az 1.3. ábrán látható AgentID (Ügynökazonosító) mező jó példa az elsődleges kulcsra, mert egyedileg azonosítja az egyes ügynököket az Agents (Ügynökök) táblán belül, és a többször szereplő rekordok kiküszöbölésével gondoskodik a táblaszintű következetességről. A kulcsot ezenkívül arra is használjuk, hogy az Agents és az adatbázis más táblái (például – ahogy az ábrán láthatjuk – az Entertainers tábla) között kapcsolatokat alakítsunk ki.



1.3. ábra

Elsődleges és idegen kulcsok

Amikor eldöntjük, hogy két táblát valamilyen módon összekapcsolunk, a kapcsolatot általában úgy hozzuk létre, hogy az első tábla elsődleges kulcsát lemásoljuk és beszúrjuk a második táblába, ahol az idegen kulcs lesz. (Az *idegen kulcs* kifejezés onnan ered, hogy a második táblának már van saját elsődleges kulcsa, és az első táblából átvett elsődleges kulcs a második tábla számára „idegen”.)

Az 1.3. ábrán egy jó példát láthatunk az idegen kulcsra. A példában az AgentID elsődleges kulcs az Agents táblában és idegen kulcs az Entertainers táblában. Amint láthatjuk, az Entertainers táblának már van egy elsődleges kulcsa – az EntertainerID. Ebben a kapcsolatban az AgentID az a mező, amely megteremti az összefüggést az Agents és az Entertainers tábla között.

Az idegen kulcsok nem csak abból a nyilvánvaló szempontból fontosak, hogy segítenek kapcsolatot teremteni két tábla között, hanem azért is, mert biztosítják a kapcsolatszintű következetességet, vagyis a két tábla rekordjai mindig helyes kapcsolatban fognak állni egymással, mivel az idegen kulcs mező értékeinek a hivatkozott elsődleges kulcs értékeiből *kell* származniuk. Az idegen kulcsok a rettegett „árva rekordokat" is segítenek elkerülni, amelyek klasszikus példája egy olyan rendelési rekord, amelyhez nem tartozik vásárló. Ha nem tudjuk, ki adta fel a rendelést, nem leszünk képesek feldolgozni azt, és természetesen kiszámlázni sem tudjuk, ami rontja a negyedéves eladási mutatóinkat.

Nézettáblák

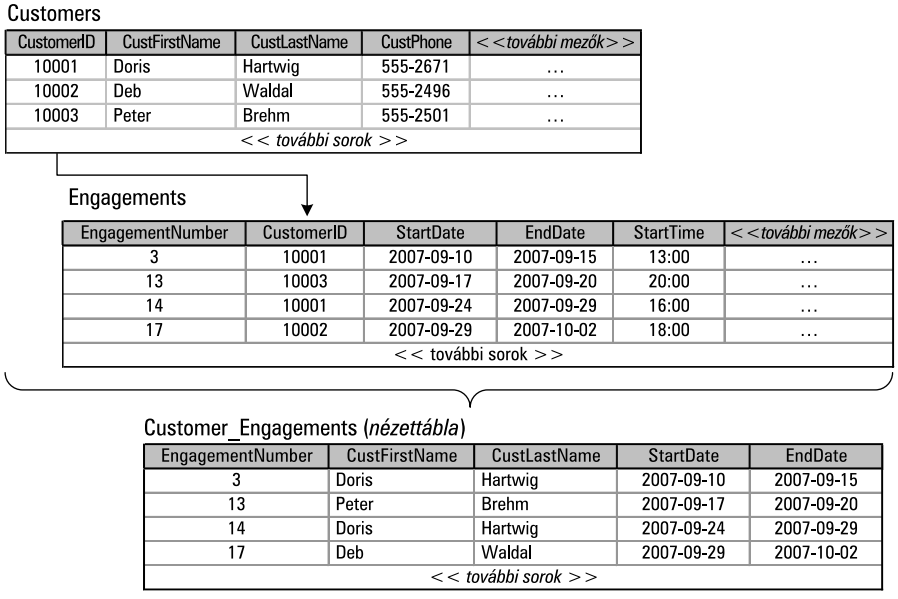
A nézet vagy nézettábla olyan virtuális tábla, amely az adatbázis egy vagy több táblájának mezőiből épül fel. A nézettáblát alkotó táblákat *alaptáblákna*k hívjuk. A relációs modell azért tekinti virtuálisnak (látszólagosnak) a nézettáblákat, mert azok adatai más táblákból származnak. Maguk a nézettáblák nem tárolnak adatokat; az adatbázisban valójában a szerkezetük az egyetlen információ, ami tárolódik róluk.

A nézettáblák lehetővé teszik, hogy az adatbázisban tárolt információkat számos különféle szempontból tekintsük meg, ami nagy rugalmasságot nyújt az adatok kezelésében. Nézettáblákat többféleképpen is létrehozhatunk – különösen akkor hasznosak, amikor több, egymással kapcsolatban álló táblára alapozzuk őket. Létrehozhatunk például egy olyan nézettáblát, amelyik olyan információkat összegez, mint a Seattle belvárosában dolgozó ácsok által ledolgozott órák teljes száma, vagy egy olyat, amelyik adott mezők szerint csoportosítja az adatokat. Ez utóbbi fajtára jó példa egy olyan nézettábla, amelyik egy adott terület államait veszi alapul, hogy megmutassa az egyes városokban az alkalmazottak teljes számát. A nézettáblák jellemző felépítését az 1.4. ábrán láthatjuk.

Sok RDBMS program a nézettáblákat *mentett lekérdezésként* valósítja meg, és ezen a néven vagy egyszerűen *lekérdezésként* hivatkozik rájuk. A lekérdezések a legtöbb esetben rendelkeznek a nézettáblák minden jellemzőjével, vagyis csupán a nevük az egyetlen különbség köztük. (Mi már gyakran morfondíroztunk rajta, hogy ehhez nem egy marketingcsapatnak van-e köze.) Fontos megjegyezni, hogy egyes gyártók már kezdik a valódi nevükön nevezni a lekérdezéseket, de mindegy, hogy a mi relációs adatbázis-kezelőnk hogy hívja őket, az adatbázisban valójában nézettáblákkal dolgozunk.

Könyvünk címe ettől függetlenül *SQL-lekérdezések földi halandóknak,* miközben elsősorban azt szeretné megtanítani, hogy miként építsük fel a nézettábláinkat. Ahogy a 2. fejezetből majd megtudhatjuk, egy relációs adatbázist akkor tervezünk meg helyesen, ha

az adatainkat úgy bontjuk fel, hogy minden tárgyhoz vagy eseményhez egyetlen tábla tartozzon. Ugyanakkor többnyire összefüggő tárgyakról vagy eseményekről szeretnénk információt szerezni – mely vásárlók és milyen rendelést adtak fel, mely tanárok milyen órákat tartanak, és így tovább –, ehhez pedig nézettáblát kell létrehoznunk, és tudnunk kell, hogy milyen SQL kóddal érhetjük ezt el.



1.4. ábra

Példa egy nézettáblára

Kapcsolatok

Ha egy adott tábla rekordjai valamilyen módon egy másik tábla rekordjaihoz társíthatók, azt mondjuk, hogy a táblák között kapcsolat áll fenn. A kapcsolat megteremtésének módja a kapcsolat típusától függ. Két tábla között háromféle kapcsolat állhat fenn: egy-az-egyhez, egy-a-többhöz (egy-a-sokhoz) vagy több-a-többhöz (sok-a-sokhoz) A kapcsolatok fajtáinak ismerete létfontosságú ahhoz, hogy megértsük, hogyan működnek a nézettáblák, és hogy miként kell a többtáblás SQL-lekérdezéseket megtervezni és használni. (Erről a III. részben tanulunk majd bővebben.)

Egy-az-egyhez

Két tábla akkor áll egy-az-egyhez kapcsolatban, ha az első tábla egy rekordja a második tábla *egyetlen* rekordjához kapcsolódik, és a második tábla egy rekordja is *egyetlen* rekorddal áll kapcsolatban az első táblából. Ebben a fajta kapcsolatban az egyik tábla *elsődleges tábla*, míg a másik *másodlagos tábla* lesz. A kapcsolatot úgy hozzuk létre, hogy vesszük

az elsődleges tábla elsődleges kulcsát, és beszúrjuk azt a másodlagos táblába, ahol idegen kulcs lesz belőle. Ez a kapcsolatok különleges fajtája, mivel sok esetben az idegen kulcs egyben a másodlagos tábla elsődleges kulcsaként is viselkedik.

Az egy-az-egyhez kapcsolatra az 1.5. ábrán láthatunk egy jellemző példát: itt az Agents (Ügynökök) az elsődleges, a Compensation (Díjazás) pedig a másodlagos tábla. A két tábla között olyan kapcsolat áll fenn, amelyben az Agents tábla egy-egy rekordja csak egy-egy rekordhoz kapcsolódik a Compensation táblában, és a Compensation tábla egy-egy rekordja is csak egy-egy rekordhoz társul az Agents táblából. Figyeljük meg, hogy az AgentID valóban elsődleges kulcs mindkét táblában, de egyben idegen kulcs a másodlagos táblában.

A kapcsolatban a vezető szerepet játszó elsődleges tábla kijelölése teljesen tetszőleges. Az egy-az-egyhez kapcsolatok nem túl gyakoriak; általában akkor találkozunk velük, ha egy táblát titkosítás végett két részre bontottak.

Agents

AgentID	AgentFirstName	AgentLastName	DateOfHire	AgentHomePhone	<<*további mezők*>>
1	William	Thompson	1997-05-15	555-2681	…
2	Scott	Bishop	1998-02-05	555-2666	…
3	Carol	Viescas	1997-11-19	555-2571	…

Compensation

Salary	CommissionRate	<<*további mezők*>>
$35,000.00	4.00%	…
$27,000.00	4.00%	…
$30,000.00	5.00%	…

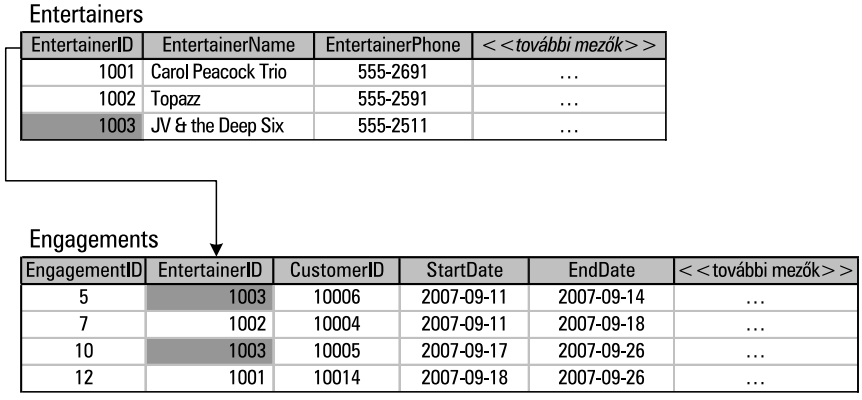
1.5. ábra

Példa egy-az-egyhez kapcsolatra

Egy-a-többhöz

Ha két tábla egy-a-többhöz kapcsolatban áll, akkor az első tábla egy rekordja a második tábla *több* rekordjához is kapcsolódhat, de a második tábla egy rekordja csak *egyetlen* rekorddal állhat kapcsolatban az első táblából. Ezt a kapcsolatot úgy hozzuk létre, hogy vesszük a kapcsolat „egy” oldalán álló tábla elsődleges kulcsát, és beszúrjuk azt a „több” oldalon levő táblába, ahol idegen kulcs lesz belőle.

Az egy-a-többhöz kapcsolatra az 1.6. ábrán láthatunk egy jellemző példát: itt az Entertainers (Előadók) tábla egy-egy rekordja *több* rekordhoz kapcsolódik az Engagements (Rendezvények) táblában, de az Engagements tábla egy-egy rekordja csak *egyetlen* rekordhoz társul az Entertainers táblából. Ahogy nyilván kitaláltuk, az Engagements tábla idegen kulcsa az EntertainerID.



Entertainers

EntertainerID	EntertainerName	EntertainerPhone	<<*további mezők*>>
1001	Carol Peacock Trio	555-2691	…
1002	Topazz	555-2591	…
1003	JV & the Deep Six	555-2511	…

Engagements

EngagementID	EntertainerID	CustomerID	StartDate	EndDate	<<további mezők>>
5	1003	10006	2007-09-11	2007-09-14	…
7	1002	10004	2007-09-11	2007-09-18	…
10	1003	10005	2007-09-17	2007-09-26	…
12	1001	10014	2007-09-18	2007-09-26	…

1.6. ábra

Példa egy-a-többhöz kapcsolatra

Több-a-többhöz

Két tábla akkor áll több-a-többhöz kapcsolatban, ha az első tábla egy rekordja a második tábla *több* rekordjához kapcsolódik, és a második tábla egy rekordja is *több* rekorddal áll kapcsolatban az első táblából. Ezt a kapcsolatot helyesen úgy hozzuk létre, hogy létrehozunk egy úgynevezett *kapcsoló táblát*. Ennek a táblának a segítségével egyszerűen társíthatunk rekordokat az egyik táblából a másik tábla rekordjaihoz, és a kapcsoló tábla arról is gondoskodik, hogy a kapcsolódó adatok hozzáadása, törlése vagy módosítása során semmilyen gond ne lépjen fel. A kapcsoló táblát úgy határozzuk meg, hogy lemásoljuk a kapcsolatban részt vevő mindkét tábla elsődleges kulcsát, és ezek segítségével alkotjuk meg az új tábla szerkezetét. A kulcsmezők két jól elkülöníthető szerepet töltenek be: együttesen a kapcsoló tábla összetett elsődleges kulcsát adják, külön-külön pedig idegen kulcsokként viselkednek.

Customers

CustomerID	CustFirstName	CustLastName	CustPhone	<<*további mezők*>>
10001	Doris	Hartwig	555-2671	…
10002	Deb	Waldal	555-2496	…
10003	Peter	Brehm	555-2501	…

Entertainers

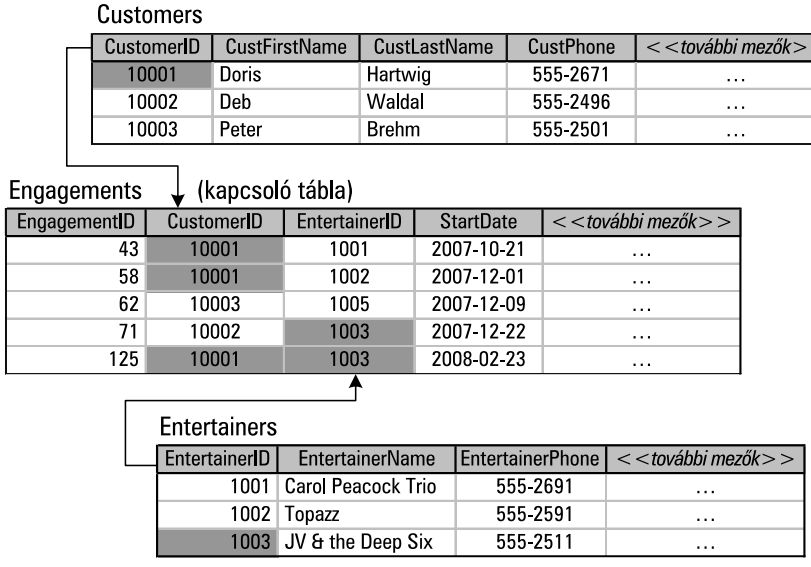
EntertainerID	EntertainerName	EntertainerPhone	<<*további mezők*>>
1001	Carol Peacock Trio	555-2691	…
1002	Topazz	555-2591	…
1003	JV & the Deep Six	555-2511	…

1.7. ábra

Feloldatlan több-a-többhöz kapcsolat

Ha egy több-a-többhöz kapcsolat nem megfelelően jött létre, *feloldatlan kapcsolatról* beszélünk. Erre az 1.7. ábrán láthatunk egy világos példát. Itt a Customers (Megrendelők) tábla egy-egy rekordja az Entertainers tábla *több* rekordjához kapcsolódhat, és az Entertainers tábla egy-egy rekordja is a Customers tábla *több* rekordjához.

A kapcsolat a több-a-többhöz viszonyban rejlő probléma miatt lesz feloldatlan. A kérdés ez: miként társíthatjuk egyszerűen az első tábla rekordjait a második tábla rekordjaihoz? Ha a kérdést az 1.7. ábrán látható táblák szerint fogalmazzuk újra, a probléma ez lesz: hogyan társíthatunk egy megrendelőt több előadóhoz vagy egy adott előadót több megrendelőhöz? (Ha egy szórakoztatóügynökséget vezetünk, bizonyára azt reméljük, hogy idővel minden megrendelő több előadót is leköt, illetve az egyes előadók fellépéseire is többen lesznek kíváncsiak.) Beszúrjuk pár megrendelő mezőjét az Entertainers táblába? Vagy adjunk néhány előadómezőt a Customers táblához? Mindkét megoldás számos problémát eredményez, amikor a kapcsolódó adatokkal dolgozunk, elsősorban az adatok következetessége terén. A problémára az a megoldás, hogy a korábban ismertetett módon létrehozunk egy kapcsoló táblát, amellyel megfelelően feloldhatjuk a több-a-többhöz kapcsolatot. A megoldás gyakorlati megvalósítását az 1.8. ábra mutatja.



1.8. ábra

Megfelelően feloldott több-a-többhöz kapcsolat

Az 1.8. ábrán a kapcsoló táblát úgy hoztuk létre, hogy vettük a CustomerID mezőt a Customers táblából, valamint az EntertainerID mezőt az Entertainers táblából, és ezekből alkottuk meg az új táblát. Az adatbázis más tábláihoz hasonlóan a kapcsoló táblának is saját ne-

ve van: Engagements (Rendezvények). Az Engagements tábla jó példa egy olyan táblára, amely egy eseményről tárol információkat. Láthatjuk például, hogy az 1003-as előadó (JV & the Deep Six) a 10001-es megrendelőnél (Doris Hartwig) lépett fel február 23-án. A kapcsoló tábla igazi előnye, hogy lehetővé teszi, hogy a kapcsolatban részt vevő mindkét táblából tetszőleges számú rekord között teremtsünk kapcsolatot. Ahogy a példa is mutatja, most már egyszerűen összekapcsolhatunk egy adott megrendelőt tetszőleges előadókkal vagy egy adott előadót akárhány megrendelővel.

Ahogy korábban mondtuk, a kapcsolatok megértése nagyon fontos ahhoz, hogy hatékony többtáblás SQL-lekérdezéseket írjunk, ezért amikor elérünk a könyv III. részéhez, érdemes visszalapozni ide, és felfrissíteni az emlékezetünket.

Miért hasznosak számunkra a relációs adatbázisok?

De miért is kell törődnünk azzal, hogy megértsük a relációs adatbázisokat? Miért érdekeljen minket, hogy milyen környezetben kezeljük az adatainkat? És egyáltalán: miért hasznosak számunkra a relációs adatbázisok? Nos, ideje, hogy megvilágosítsuk az elménket, és fejest ugorjunk a mélyvízbe!

Az idő, amit a relációs adatbázisok tanulmányozására fordítunk, olyan befektetés, ami egyértelműen megtérül. Aktív tudásra kell szert tennünk a relációs adatbázisokkal kapcsolatban, mivel ma ez a legszélesebb körben elterjedt adattárolási modell. Felejtsük el, amit korábban olvastunk a témáról vagy amit Huba az informatikai részlegről mondott nekünk – a cégek és különféle szervezetek által gyűjtött és kezelt adatok túlnyomó többségét relációs adatbázisokban tárolják. A modellnek valóban léteznek bővítései, a relációs adatbázisokat kezelő programok ma már objektumközpontú megoldásokat is alkalmaznak, és az ilyen adatbázisok mára teljesen beépültek a Világháló szövetébe, de nem számít, hogy csűrjük-csavarjuk, aprítjuk és fűszerezzük, ezek az adatbázisok ettől még relációs adatbázisok maradnak. A relációs modell 35 éve velünk él, a szerepe csak egyre erősödik, és belátható időn belül nem lesz, ami felváltsa.

Szinte minden ma használt kereskedelmi adatbázis-kezelő szoftver relációs alapokon nyugszik (bár Dr. Codd, C.J. Date és Fabian Pascal komolyan megkérdőjeleznék, hogy bármelyik kereskedelmi megvalósítás valóban relációs-e!). Ha adatbázisokkal dolgozunk, és sikert szeretnénk elérni, tudnunk kell, hogyan kell megtervezni egy relációs adatbázist, és hogyan kell azt megvalósítani valamelyik népszerű RDBMS programban – ráadásul azóta, hogy oly sok cég folytat üzleti tevékenységet az Interneten, némi webfejlesztési tapasztalat sem árt.

A relációs adatbázisok gyakorlati ismerete több szempontból is a segítségünkre lehet. Minél többet tudunk például a relációs adatbázisok tervezéséről, annál könnyebben tudunk majd egy adott adatbázishoz felhasználói alkalmazásokat fejleszteni. Meg fogunk lepődni, milyen

egyszerűvé válik az RDBMS programunk használata – mivel értjük, hogy mire szolgálnak az egyes eszközei, és hogyan használhatjuk azokat a leghatékonyabban. Gyakorlati ismereteinknek akkor is jó hasznát vesszük, amikor az SQL használatát igyekszünk elsajátítani, hiszen az SQL a relációs adatbázisok létrehozásának és kezelésének szabványos nyelve.

Hogyan tovább?

Már értjük, hogy miért fontos megismerkednünk a relációs adatbázisokkal, de azt is látnunk kell, hogy az *adatbázisok elmélete* és az *adatbázisok tervezése* két különböző dolog. Az adatbázisok elméletét azok az elvek és szabályok alkotják, amelyeken a relációs adatbázismodell alapul: ez az, amit az iskolák szentélyeiben oktatnak, és amit a valóság sötét bugyraiban gyorsan el kell hajítanunk. Az elmélet persze fontos, mert nélküle nem készíthetnénk szerkezetileg helyes relációs adatbázisokat, és az adatbázisaink adatain végzett műveletek eredménye sem lenne megjósolható. Ezzel szemben viszont az adatbázisok tervezése azokat a strukturált folyamatokat foglalja magába, amelyek révén egy relációs adatbázis felépül. A helyes adatbázis-tervezési módszerek ismerete elengedhetetlen ahhoz, hogy pontos és következetes adatokat tartalmazó adatbázisokat hozzunk létre, illetve hogy a kinyert információk is a lehető legpontosabbak és legfrissebbek legyenek.

Ha teljes vállalatra kiterjedő adatbázisokat szeretnénk tervezni és létrehozni, webes alapú internetes üzleti adatbázisokat akarunk fejleszteni, vagy adattárakat kívánunk üzemeltetni, célszerű megfontolni az adatbázisok elméletének tanulmányozását. Ez akkor is érvényes, ha az említett területek egyikében sem akarunk elmerülni, csak elismert adatbázis-szakértővé szeretnénk válni. Mindenki másnak, aki relációs adatbázisokat tervez és készít különböző rendszerekre (és úgy véljük, ők teszik ki e könyv olvasóinak nagy többségét), a helyes adatbázis-tervezési módszerek alapos ismerete elegendő. Ne feledjük, hogy egy adatbázist megtervezni viszonylag könnyű, de *megvalósítani* azt egy adott RDBMS programban egy adott rendszerre már teljesen más tészta (amit más könyvekből sajátíthatunk el).

A piacon számos jó könyvet találunk az adatbázis-tervezésről. Egyesek, például Mike Hernandez *Database Design for Mere Mortals* című könyve (Addison-Wesley, 2004; magyarul: *Adatbázis-tervezés: A relációs adatbázisok alapjairól földi halandóknak*. Kiskapu, 2004) csak az adatbázis-tervezés módszertanával foglalkoznak, míg mások, például C.J. Date *An Introduction to Database Systems* (Addison-Wesley, 2003) című műve, keverik az elméletet és a gyakorlati tervezést. (Arra viszont fel kell hívnunk a figyelmet, hogy az elméleti könyvek általában nem könnyű olvasmányok.) Miután eldöntöttük, hogy milyen irányban szeretnénk továbbhaladni, válasszuk ki és vegyük meg a megfelelő könyveket, fogjunk egy csésze dupla eszpresszót (vagy ha más kedvenc italunk van, akkor azt), és merüljünk el bennük. Ha már általánosságban magabiztosan mozgunk a relációs adatbázisok körében, rá fogunk jönni, hogy mélyebben meg kell ismerkednünk az SQL-lel is – ezért olvassuk ezt a könyvet.

Összefoglalás

A fejezetet a ma használatos adatbázisok különböző típusainak rövid ismertetésével kezdtük. Megtudtuk, hogy a dinamikus adatokkal dolgozó vállalatok és intézmények műveleti adatbázisokat használnak, és így biztosítják, hogy a kinyert információk mindig a lehető legfrissebbek és legpontosabbak legyenek, míg a statikus adatokat kezelő szervezetek elemző adatbázisokat alkalmaznak.

Ez után röviden áttekintettük a relációs adatbázismodell történetét. Elmeséltük, hogy a modellt Dr. E. F. Codd alkotta meg a matematika egyes ágaira alapozva, és hogy a modell már több mint 35 éve létezik. Adatbázis-kezelő programokat, ahogy ma ismerjük őket, különféle számítógépes környezetekhez készítenek, a teljesítményük, a hatékonyságuk és a képességeik köre pedig az 1970-es évek óta folyamatosan nő. A nagygépektől az asztali és webes alkalmazásokig sok-sok vállalat gerincét RDBMS programok képezik.

Ezt követően a relációs adatbázisok felépítését vizsgáltuk. Bemutattuk az alapvető összetevőiket, és röviden elmagyaráztuk a szerepüket. Tanultunk a kapcsolatok három lehetséges típusáról, és megértettük, miért fontosak, nem csak magának az adatbázis szerkezetének a szempontjából, hanem abból a szempontból is, hogy miként segítik az SQL működésének jobb megértését.

Végül elmagyaráztuk, hogy milyen hasznunk származik a relációs adatbázisok tanulmányozásából, és hogy miként kell megtervezni őket. Most már tudjuk, hogy a ma használatos adatbázisok közül a relációs adatbázisok a legelterjedtebbek, és szinte minden adatbázis-kezelő szoftver mögött, amivel találkozhatunk, egy relációs adatbázis áll. Ezen kívül arról is képet kaptunk, hogy merre haladhatunk, ha kicsit többet szeretnénk tudni a relációs adatbázisok elméletéről és tervezéséről.

A következő fejezetben arra tanulunk meg néhány módszert, hogy miként finomhangolhatunk meglevő adatbázis-szerkezeteket.