“Capítol 4: SQL Intermedi”

Fitxers i bases de dades

Capítol 4: SQL Intermedi

- Expressions Join
- Views
- Transaccions
- Restriccions d'Integritat
- Tipus de dades i esquemes SQL
- Autorització

Relacions Joined

- Operacions Join pren dues relacions i retorna com a resultat una altra relació
- Una operació Join és un producte Cartesià el qual requereix que les tuples a les dues relacions combinin (sota alguna condició). També especifica els atributs que són presents en el resultat de la unió
- Les operacions Join són típiques en les expressions de subconsultes de la condició **from**

Operacions join - Exemple

- Relació *course*

course_id	title	dept_name	credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relació *prereq*

course_id	prereq_id
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

- Observeu que
  - La informació prereq és missing per CS-315 i
  - La informació del curs per CS-347 és missing

Outer Join

- Una extensió de l'operació join que evita la pèrdua d'informació.
- Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.
- Utilitza valors *null*.

Left Outer Join

- *course* **natural left outer join** *prereq*

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*

Right Outer Join

- *course* **natural right outer join** *prereq*

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101

Relacions Joined

- Operacions Join pren dues relacions i retorna com a resultat una altra relació
- Aquestes operacions addicionals són típiques en expressions de subconsultes a la condició **from**
- Condicions Join - defineix quines tuple de les dues relacions són les que concorden, i quins atributs es conserven en el resultat del join.
- Join type - defineix què es fa amb les tuple que no tenen correspondència en l'altra relació.

Join types
inner join
left outer join
right outer join
full outer join

Join Conditions
natural
on <predicate>
using $(A_1, A_1, \ldots, A_n)$

Figure: Exemple de relacions joined.

Full Outer Join

- *course* **natural full outer join** *prereq*

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101

Joined Relations - Example

- *course* **inner join** *prereq* **on**

  course.course_id = *prereq.course_id*

course_id	title	dept_name	credits	prereq_id	course_id
BIO-301	Genetics	Biology	4	BIO-101	BIO-301
CS-190	Game Design	Comp. Sci.	4	CS-101	CS-190

- Quina és la diferència entre la taula anterior i un natural join?
- *course* **left outer join** *prereq* **on**

  course.course_id = *prereq.course_id*

course_id	title	dept_name	credits	prereq_id	course_id
BIO-301	Genetics	Biology	4	BIO-101	BIO-301
CS-190	Game Design	Comp. Sci.	4	CS-101	CS-190
CS-315	Robotics	Comp. Sci.	3	*null*	*null*

Joined Relations - Example

- *course* **natural right outer join** *prereq*

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101

- *course* **full outer join** *prereq* **using** (*course_id*)

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101

Views

- En alguns casos, no és desitjable per a tots els usuaris veure el mòdel lògic sencer (això és, totes les relacions existents desades a la base de dades).
- Considera una persona que necessita saber el nom d'un instructor i el seu departament, però no el seu salari. Aquesta persona hauria de veure una relació descrita, en SQL, per

  select *ID, name, dept_name*

  from *instructor*

- Una **view** proporciona un mecanisme per amagar certes dades des de la vista de certs usuaris.
- Qualsevol relació que no és del model conceptual però és visible a qualsevol usuari com a "relació virtual" és anomenada una **view o vista**

Definició de Vista

- Una vista es defineix amb la sentència **create view** que té la forma

  create view *v* **as** <expressió de búsqueda>

  on <expressió de búsqueda> és qualsevol expressió legal d'SQL. El nom de la vista és representat per *v*.

- Un cop una vista està definida, el seu nom es pot fer servir per referir-se a la relació virtual que ha creat

- La definició d'una vista no és la mateixa que crear una nova relació fent servir l'expressió de cerca

  - La definició d'una vista provoca guardar una expressió; l'expressió és substituïda en cerques fent servir la vista.

Exemples de Vistes

- Una vista d'instructors sense el seu salari

  create view *faculty* **as**
  select *ID*, *name*, *dept_name*
  from *instructor*

- Trobar tots els instructors del departament de Biologia

  select *name*
  from *faculty*
  where *dept_name* = 'Biology'

- Crear una vista amb els salaris de tots els departaments

  create view *departments_total_salary(dept_name, total_salary)* **as**
  select *dept_name*, **sum**(*salary*)
  from *instructor*
  grouped by *dept_name*;

Vistes definides utilitzant altres vistes

- **create view** physics_fall_2009 **as**

  select course.course_id, sec_id, building, room_number
  from *course, section*
  where *course.course_id* = textitsection.course_id
  and *course.dept_name* = 'Physics'
  and *section.semester* = 'Fall'
  and *section.year* = '2009'

- **create view** physics_fall_2009_watson **as**

  select *course_id, room_number*
  from *physics_fall_2009*
  where *building* = 'Watson';

View Expansion

- Estirar l'ús d'una view en una query/altra view

create view *physics_fall_2009_watson* **as**
(**select** *course_id*, *room_number*
from (**select** *course.course_id*, *building*, *room_number*
from *course*, *section*
where *course.course_id* = *section.course_id*
and *course.dept_name* = 'Physics'
and *section.semester* = 'Fall'
and *section.year* = '2009')
where *building* = 'Watson';)

Views definides utilitzant altres views

- Una vista pot ser utilitzada en l'expressió definint una altra vista
- Una relació de vista v_1 s'anomena directament dependent sobre una relació de vista v_2 si v_2 és utilitzat en l'expressió definint v_1
- Una relació de vista v_1 s'anomena dependent sobre una relació de vista v_2 si v_1 depèn directament de v_2 o hi ha un camí de dependències entre v_1 i v_2
- Una relació de vista v s'anomena recursiva si depèn d'ella mateixa.

View Expansion

- Una manera de definir el significat de les vistes definides en termes d'altres vistes.
- Sigui la vista v_1 definida per una expressió e_1 que pot contenir ella mateixa usos de relacions de vistes.
- L'expansió de vista d'una expressió segueix els següents passos:

  repeat

  Trobar qualsevol relació de vista v_i en e_1

  Substitueix la relació de vista v_i per l'expressió definidia v_i

  until No hi ha més relacions de vista en e_1

- Aquest bucle acabarà sempre i quan la definició de la vista no sigui recursiva.

Actualitzar una vista

- Afegir una nova tuple a *faculty* definida anteriorment

  insert into *faculty* **values** ('30765', 'Green', 'Music');

  Aquesta introducció ha de ser representada per la introducció de la tuple ('30765', 'Green', 'Music', null)

  dins de la relació *instructor*

Algunes actualitzacions no són explicades de manera única

- **create view** *instructor_info* **as**

  select *ID, name, building*

  from *instructor, department*

  where *instructor.dept_name* = *department.dept_name*

- **insert into** *instructor_info* **values** ('69987', 'White', 'Taylor');
  - Quin departament, si n'hi ha de múltiples en Taylor?
  - Què passa si no hi ha departament en Taylor?
- Moltes de les implementacions d'SQL permeten actualitzacions només en vistes simples
  - La condició **from** només té una relació base de dades.
  - la condició **select** conté només noms d'atributs de la relació, i no té cap expressió, agregat o especificació **distinct**.
  - Qualsevol atribut no llistat amb la condició **select** pot ser fixada com a null.
  - La consulta no té una condició **group by** o **having**.

And Some Not at All

- **create view** *history_instructors* **as**

  select *

  from *instructor*

  where *dept_name* = 'History'

- Què passa si s'introdueix ('25566', 'Brown', 'Biology', 100000) dins d'*history_instructors*?

Materialized Views

- **Materialitzant una vista**: Crear una taula física contenint totes les tuples en el resultat de la consulta definint la vista.
- Si la relació feta servir en la consulta està actualitzada, el resultat de la vista materialitzada es converteix en obsoleta
  - Necessitat de mantenir la vista, actualitzant-la cada cop que les relacions per sota són actualitzades

Transaccions

- Unitat de treball
- Transacció atòmica
  - ja sigui totalment executada o es desfà com si mai hagués passat
- Aïllament de transaccions concurrents
- Transaccions comencen implícitament
  - Acabades per **commit work** o **rollback work**
- Però per defecte a la majoria de bases de dades: cada expressió d'SQL encarrega automàticament
  - Pot apagar l'auto commit per una sessió (Ex.: fent servir API)
  - En SQL:1999, es pot fer servir: **begin automatic** . . . **end**
    - No permès en la majoria de bases de dades

Restriccions d'integritat

- Les restriccions d'integritat protegeixen contra danys accidentals a la base de dades, assegurant que els canvis autoritzats no resulten en una pèrdua de la consistència de les dades.
  - Un compte de xecs ha de tenir un balanç més gran de 10.000,00$
  - El salari d'un treballador de banca ha de ser, com a mínim, de 4.00$ l'hora
  - Un client ha de tenir un número de telèfon (no nul)

Restriccions d'integritat en una Relació Simple

- **not null**
- **primary key**
- **unique**
- **check**(P), on P és la declaració

Restriccions Not Null i Unique

- **not null**
  - Declarar que *name* i *budget* siguin **not null**

    name **varchar**(20) **not null**
    budget **numeric**(12,2) **not null**

- **unique** $(A_1, A_2, \ldots, A_m)$
  - L'única especificació manifesta que els atributs $A_1, A_2, \ldots, A_m$ formen una candidata a clau.
  - Les candidates a claus poden ser nul·les (a restriccions de claus primàries)

La condició Check

- **check** (P)

  On P és una declaració

  Exemple: Garantir que el semestre és un dels quatre següents: Tardor, hivern, primavera o estiu:

```
create table section(
    course_id varchar(8),
    sec_id varchar(8),
    semester varchar(4,0),
    year numeric(8),
    building varchar(15),
    room_number varchar(7),
    time slot id varchar(4),
    primary key course_id, sec_id, semester, year),
    check (semester in ('Fall', 'Winter', 'Spring', 'Summer'))
);
```

Integritat Referencial

- Assegura que un valor que apareix en una relació un un conjunt donat d'atributs també apareix per a cert conjunt d'atributs en una altra relació.
    - Exemple: Si "Biology" és un departament on el seu nom apareix en una tuple en la relació *instructor*, aleshores existeix una tuple en la relació *department* per "Biology".
- Sigui A un conjunt d'atributs. Siguin R i S dues relacions que conteten atributs A on A és la clau primària de S. A s'anomena foreign key d'R si per valors d'A que apareguin a R aquests valors també apareixen a S.

Accions Cascada en Integritat Referencial

- **create table** *course*(
  course_id **char**(5) **primary key**,
  title **varchar**(20),
  dept_name **varchar**(20) **references** *department*
)
- **create table** *course*(
  ...
  dept_name **varchar**(20),
  foreign key(*dept_name*) **references** *department*
  on delete cascade
  on update cascade
  ...
)
- Accions alternatives a cascades: **set null, set default**

Violació de la Restricció d'Integritat durant Transaccions

- Exemple:

  create table *person*(
  ID **char**(10),
  name **char**(40),
  mother **char**(10),
  father **char**(10),
  foreign key *father* **references** *person*,
  foreign key *mother* **references** *person*)

- Com insertar una tuple sense generar violar restriccions?
  - Agefir father i mother d'una persona abans d'afegir la persona
  - O, fixar father i mother com a null incialment, actualitzar després d'insertar totes les persones (no serà possible si els atributs father i mother són declarats **not null**
  - O aplaçar la validació de la restricció (pròxima transparència)

Condicions Check Complexes

- **check** (*time_slot_id* **in** (**select** *time_slot_id* **from** *time_slot*))
  - Per què no utilitzar una foreign key?
- Cada secció té com a mínim un instructor ensenyant la secció.
  - Com s'escriu això?
- Per desgràcia: subconsultes en condicions check no suportades la majoria de les vegades
  - Alternativa: triggers (més endavant)
- **create assertion** <assertion-name> **check** <predicate>;
  - També no suportat per qualsevol

Build-in Data Types in SQL

- data: dates, que continguin un any (4 dígits), mes i dia
  - Exemple: **date** '2005-7-27'
- time: temps del dia, en hores, minuts i segons
  - Exemple: **time** '09:00:30' **time** '09:00:30.75'
- timestamp: data i temps del dia
  - Exemple: **timestamp** '2005-7-27 09:00:30.75'
- interval: període de temps
  - Exemple: interval '1' dia
  - Extraient un valor data/temps/timestamp d'un altre dóna un valor interval
  - Valors intervals poden ser afegides a valors data/temps/timestamp

Index Creation

- **create table** *student*

  (*ID* **varchar**(5),

  name **varchar**(20) **not null**,

  dept_name **varchar**(20),

  tot_cred **numeric**(3,0) **default** 0,

  primary key (*ID*))

- **create index** *studentID_index* **on** *student(ID)*

- Els índexs són estructures de dades utilitzades per accelerar l'accés als registres amb valors especificats per atributs indexats

  - Ex.: **select** *

    from *student*

    where *ID* = '12345'

    pot ser executat fent servir l'índex per trobar el registre necessari, sense buscar a tots els registres d'*student*

    Més sobre índexs al capítol 11

User-Defined Types

- create type construct in SQL creates user-defined type

  create type *Dollars* **as numeric (12,2) final**

  - **create type** *department*
    (*dept_name* **varchar**(20),
    building **varchar**(15),
    budget Dollars);

Dominis

- create domain construct in SQL-92 creates user-defined domain type

  create domain *person_name* **char**(20) **not null**

- Types i dominis són similars. Els dominis poden tenir restriccions, tals com **not null**, especificades sobre ells.

- **create domain** *degree_level* **varchar**(10)
  constraint *degree_level_test*
  check (**value in** ('Bachelors', 'Masters', 'Doctorate'));

Large-Objects Types

- Objectes grans (fotos, videos, fitxers CAD, etc.) són emmagatzemats com a *objectes grans*:
  - blob: Objecte gran binari - L'objecte és una col·lecció gran de dades binàries no interpretades (la seva interpretació recau en una aplicació aliena al sistema de la base de dades)
  - clob: Objecte gran caracter - L'objecte és una col·lecció gran de dades caracter
  - Quan una consulta retorna un objecte gran, en realitat es retorna **un punter** en lloc de l'objecte gran en sí mateix

Autorització

Formes d'autorització de la base de dades:

- Read - Permet la lectura, però no la modificació de dades.
- Insert - Permet l'addició de noves dades, però no la modificació de dades existents.
- Update - Permet la modificació, però no l'eliminació de dades.
- Delete - Permet l'eliminació de dades.

Formes d'autorització per modificar esquemes de la base de dades:

- Index - Permet **la creació i eliminació d'índexs**.
- Resources - Permet **la creació de noves relacions**.
- Alteration - Permet **l'addició o eliminació d'atributs en una relació**.
- Drop - Permet **l'eliminació de relacions**.

Especificació d'autorització en SQL

- La sentència grant es fa servir per concedir autorització

  grant <llista de privilegis>

  on <nom de la vista o relació> **to** <llista d'usuaris>

- La <llista d'usuaris> és:
  - un user-id
  - **public**, que permet a tots els usuaris vàlids el privilegi fet
  - una regla (per més endavant)

- Donar privilegis a una vista no implica donar cap privilegi a les relacions subjacents.

- The grantor of the privilege must already hold the privilege on the specified item (or be the database administrator)

Privilegis en SQL

- select: permet llegir l'accés a la relació, o l'habilitat de cercar fent servir vistes
  - Exemple: usuaris donats U_1, U_2 i U_3 **select** autorització sobre la relació *instructor*

    grant select on *instructor* **to** U_1, U_2, U_3
- insert: l'habilitat d'inserir tuples
- update: l'habilitat d'actualitzar fent servir la sentència update d'SQL
- delete: l'habilitat d'eliminar tuples
- all privileges: utilitzada com a forma curta per tots els possibles privilegis

Revocant Autorització en SQL

- La sentència revoke s'utilitza per revocar autorització.

  revoke <llista de privilegis>

  on <nom de la vista o relació> **from** <llista d'usuaris>

- Exemple:

  revoke select on *branch* **from** U_1, U_2, U_3

- La <llista de privilegis> pot ser **all** per revocar tots els privilegis.

- Si <revoke-list> inclou **public**, tots els usuaris perdran el privilegi excepte aquells que s'especifiqui.

- Si el mateix privilegi va ser concedit dues vegades al mateix usuari amb diferents permisos, l'usuari pot mantenir el privilegi després de la revocació.

- Tots els privilegis que depenen del privilegi revocat, també seran revocats.

Rules

- create role instructor;
- **grant** *instructor* **to Amit**;
- Els privilegis poden ser concedits a normes:
  - **grant select on** *takes* **to** *instructor*;
- Les normes poden ser concedides als usuaris, com les altres normes
  - **create role** *teaching_assistant*
  - **grant** *teaching_assistant* **to** *instructor*;
    - *instructor* hereda tots els privilegis de *teaching_assistant*
- Cadena de normes
  - **create role** *dean*;
  - **grant** *instructor* **to** *dean*;
  - **grant** *dean* **to** Satoshi;

Autorització a Vistes

- **create view** *geo_instructor* **as**
  (**select** *
  from *instructor*
  where *dept_name* = 'Geology');
- **grant select on** *geo_instructor* **to** *geo_staff*
- Suppose that a *geo_staff* member issues
  - **select** *
    from *geo_instructor*;
- Què passa si
  - *geo_staff* no té permisos en *instructor*?
  - creator of view did not have some permissions on *instructor*?

Altres autoritzacions

- **references** privilege to create foreign key
  - **grant reference** *dept_name* **on** *department* **to** Mariano;
  - Per què és necessari?
- Transferència de privilegis
  - **grant select on** *department* **to** Amit **with grant option**;
  - **revoke select on** *department* **to** Amit, Satoshi **cascade**;
  - **revoke select on** *department* **to** Amit, Satoshi **restrict**;
- Per més detalls, llegir la Secció 4.6

FINAL DEL CAPÍTOL 4

Figures

ID	*name*	*dept_name*	*tot_cred*
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

Figure: 4.01.

Figures

ID	*course_id*	*sec_id*	*semester*	*year*	*grade*
00128	CS-101	1	Fall	2009	A
00128	CS-347	1	Fall	2009	A-
12345	CS-101	1	Fall	2009	C
12345	CS-190	2	Spring	2009	A
12345	CS-315	1	Spring	2010	A
12345	CS-347	1	Fall	2009	A
19991	HIS-351	1	Spring	2010	B
23121	FIN-201	1	Spring	2010	C+
44553	PHY-101	1	Fall	2009	B-
45678	CS-101	1	Fall	2009	F
45678	CS-101	1	Spring	2010	B+
45678	CS-319	1	Spring	2010	B
54321	CS-101	1	Fall	2009	A-
54321	CS-190	2	Spring	2009	B+
55739	MU-199	1	Spring	2010	A-
76543	CS-101	1	Fall	2009	A
76543	CS-319	2	Spring	2010	A
76653	EE-181	1	Spring	2009	C
98765	CS-101	1	Fall	2009	C-
98765	CS-315	1	Spring	2010	B
98988	BIO-101	1	Summer	2009	A
98988	BIO-301	1	Summer	2010	*null*

Figure: 4.02.

Figures

ID	*name*	*dept_name*	*tot_cred*	*course_id*	*sec_id*	*semester*	*year*	*grade*
00128	Zhang	Comp. Sci.	102	CS-101	1	Fall	2009	A
00128	Zhang	Comp. Sci.	102	CS-347	1	Fall	2009	A-
12345	Shankar	Comp. Sci.	32	CS-101	1	Fall	2009	C
12345	Shankar	Comp. Sci.	32	CS-190	2	Spring	2009	A
12345	Shankar	History	32	CS-315	1	Spring	2010	A
12345	Shankar	Finance	32	CS-347	1	Fall	2009	A
19991	Brandt	Music	80	HIS-351	1	Spring	2010	B
23121	Chavez	Physics	110	FIN-201	1	Spring	2010	C+
44553	Peltier	Physics	56	PHY-101	1	Fall	2009	B-
45678	Levy	Physics	46	CS-101	1	Fall	2009	F
45678	Levy	Physics	46	CS-101	1	Spring	2010	B+
45678	Levy	Physics	46	CS-319	1	Spring	2010	B
54321	Williams	Comp. Sci.	54	CS-101	1	Fall	2009	A-
54321	Williams	Comp. Sci.	54	CS-190	2	Spring	2009	B+
55739	Sanchez	Music	38	MU-199	1	Spring	2010	A-
76543	Brown	Comp. Sci.	58	CS-101	1	Fall	2009	A
76543	Brown	Comp. Sci.	58	CS-319	2	Spring	2010	A
76653	Aoi	Elec. Eng.	60	EE-181	1	Spring	2009	C
98765	Bourikas	Elec. Eng.	98	CS-101	1	Fall	2009	C-
98765	Bourikas	Elec. Eng.	98	CS-315	1	Spring	2010	B
98988	Tanaka	Biology	120	BIO-101	1	Summer	2009	A
98988	Tanaka	Biology	120	BIO-301	1	Summer	2010	*null*

Figure: 4.03.

Figures

ID	*name*	*dept_name*	*tot_cred*	*course_id*	*sec_id*	*semester*	*year*	*grade*
00128	Zhang	Comp. Sci.	102	CS-101	1	Fall	2009	A
00128	Zhang	Comp. Sci.	102	CS-347	1	Fall	2009	A-
12345	Shankar	Comp. Sci.	32	CS-101	1	Fall	2009	C
12345	Shankar	Comp. Sci.	32	CS-190	2	Spring	2009	A
12345	Shankar	History	32	CS-315	1	Spring	2010	A
12345	Shankar	Finance	32	CS-347	1	Fall	2009	A
19991	Brandt	Music	80	HIS-351	1	Spring	2010	B
23121	Chavez	Physics	110	FIN-201	1	Spring	2010	C+
44553	Peltier	Physics	56	PHY-101	1	Fall	2009	B-
45678	Levy	Physics	46	CS-101	1	Fall	2009	F
45678	Levy	Physics	46	CS-101	1	Spring	2010	B+
45678	Levy	Physics	46	CS-319	1	Spring	2010	B
54321	Williams	Comp. Sci.	54	CS-101	1	Fall	2009	A-
54321	Williams	Comp. Sci.	54	CS-190	2	Spring	2009	B+
55739	Sanchez	Music	38	MU-199	1	Spring	2010	A-
70557	Snow	Physics	0	*null*	*null*	*null*	*null*	*null*
76543	Brown	Comp. Sci.	58	CS-101	1	Fall	2009	A
76543	Brown	Comp. Sci.	58	CS-319	2	Spring	2010	A
76653	Aoi	Elec. Eng.	60	EE-181	1	Spring	2009	C
98765	Bourikas	Elec. Eng.	98	CS-101	1	Fall	2009	C-
98765	Bourikas	Elec. Eng.	98	CS-315	1	Spring	2010	B
98988	Tanaka	Biology	120	BIO-101	1	Summer	2009	A
98988	Tanaka	Biology	120	BIO-301	1	Summer	2010	*null*

Figure: 4.04.

Figures

ID	*course_id*	*sec_id*	*semester*	*year*	*grade*	*name*	*dept_name*	*tot_cred*
00128	CS-101	1	Fall	2009	A	Zhang	Comp. Sci.	102
00128	CS-347	1	Fall	2009	A-	Zhang	Comp. Sci.	102
12345	CS-101	1	Fall	2009	C	Shankar	Comp. Sci.	32
12345	CS-190	2	Spring	2009	A	Shankar	Comp. Sci.	32
12345	CS-315	1	Spring	2010	A	Shankar	History	32
12345	CS-347	1	Fall	2009	A	Shankar	Finance	32
19991	HIS-351	1	Spring	2010	B	Brandt	Music	80
23121	FIN-201	1	Spring	2010	C+	Chavez	Physics	110
44553	PHY-101	1	Fall	2009	B-	Peltier	Physics	56
45678	CS-101	1	Fall	2009	F	Levy	Physics	46
45678	CS-101	1	Spring	2010	B+	Levy	Physics	46
45678	CS-319	1	Spring	2010	B	Levy	Physics	46
54321	CS-101	1	Fall	2009	A-	Williams	Comp. Sci.	54
54321	CS-190	2	Spring	2009	B+	Williams	Comp. Sci.	54
55739	MU-199	1	Spring	2010	A-	Sanchez	Music	38
70557	*null*	*null*	*null*	*null*	*null*	Snow	Physics	0
76543	CS-101	1	Fall	2009	A	Brown	Comp. Sci.	58
76543	CS-319	2	Spring	2010	A	Brown	Comp. Sci.	58
76653	EE-181	1	Spring	2009	C	Aoi	Elec. Eng.	60
98765	CS-101	1	Fall	2009	C-	Bourikas	Elec. Eng.	98
98765	CS-315	1	Spring	2010	B	Bourikas	Elec. Eng.	98
98988	BIO-101	1	Summer	2009	A	Tanaka	Biology	120
98988	BIO-301	1	Summer	2010	*null*	Tanaka	Biology	120

Figure: 4.05.

Figures

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
69987	White	*null*	*null*

instructor

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000
null	Taylor	*null*

department

Figure: 4.06.

Figures

Join types
inner join
left outer join
right outer join
full outer join

Join conditions
natural
on <predicate>
using $(A_1, A_2, \ldots, A_n)$

Figure: 4.07.

Figures

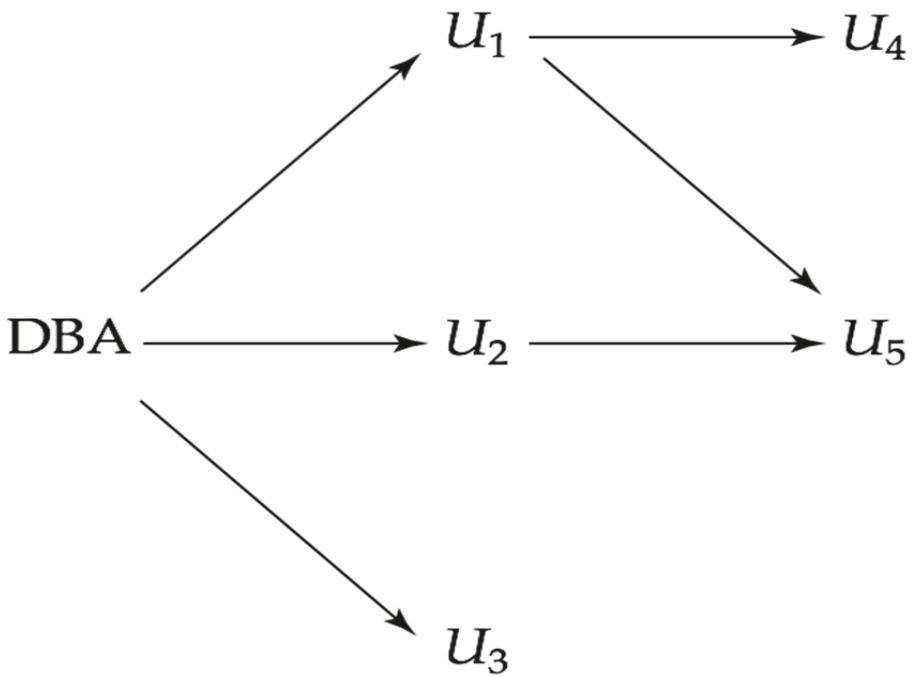

Figure: 4.08.