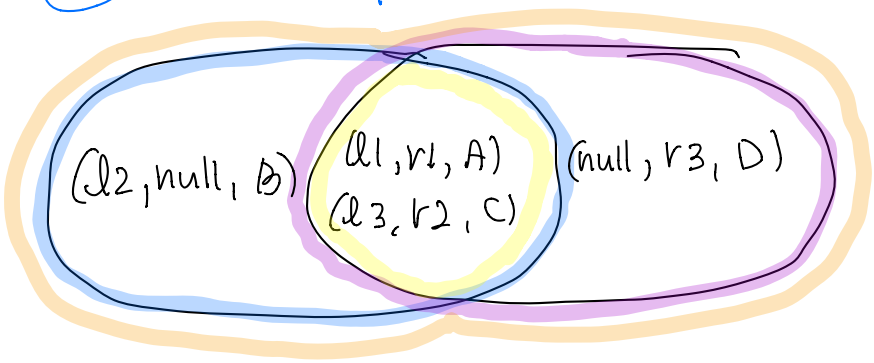
Chapter 4. Intermediate SQL

4.1 Join Expressions



(INNER) JOIN

LEFT OUTER JOIN

RIGHT OUTER JOIN

FULL OUTER JOIN

Left-hand-side relation r_l

PK	attr
l1	A
l2	B
l3	C

right-hand-side relation r_r

PK	attr
r1	A
r2	C
r3	D

ON $r_l.attr = r_r.attr$

Join predicate.

: Can include attributes that are excluded by natural join.

Remark. ON vs. WHERE

Not only ON is more human-readable in join queries, but also it behaves differently except on INNER JOIN.

Other joins with ON add null-padded tuples, like (l2, null, B), (null, r3, D) in above example, but WHERE does not until explicitly specified.

4.2 Views

SQL allows views, or "virtual relations" to be defined by a query: it is not precomputed and stored, as it is not a part of logical model, but computed on the fly, as relation is conceptually stored.

Remark. It is not desirable for all users to see the entire logic model, i.e., some attribute shouldn't be visible to all users.

Remark. Generally, it is a bad idea to compute and store query results in a seperate table, as it may become out-of-date.

Def. Materialized view maintenance is the process of keeping the materialized view (view which is kept up-to-date) up-to-date.

Remark. Some DBMS permorm maintenance lazily, or periodically, so those views should be used with care.

* Manipulations (update, insert, delete) on views' rows.
Manipulations to the database expressed in terms of view must be translated to the actual relations in the logical model. So, generally modifications are not permitted, with some exceptions.

Def. In general, SQL views are said to be updatable, (& includes update, insert, delete.
if i) Only one relation in FROM.
ii) No expressions, aggregations, DISTINCTs in SELECT.
iii) All attributes not in SELECT has a default value or nullable.
iv) No GROUP BY or HAVING.

Remark. Additional checks on manipulating rows via views can be made with WITH CHECK OPTION when defining views.

4.3 Transactions

Def. A transaction is a sequence of query and/or update statements.

Def. Commit work commits the current transaction,
to make the updates to the database in a permanent manner.
ex) Saving a document

Def. Rollback work causes the current transaction to be rolled back.
It undoes all the intermediate updates to the database.
ex) Closing a document without saving.

Remark. By either committing or rolling back the transactions
DBMS makes transactions atomic, or indivisible.
So, in many implementations, single SQL query is taken as a single transaction,
and gets commited as soon as it is executed. (But can be turned off.)

Remark. Multiple SQL queries can be formed in a single transaction
with BEGIN ATOMIC ... END.

4.4 Integrity Constraints.

Remark. Integrity constraints ensure every changes to the database to be
consistent, in other words guards accidental damages to the database.

Remark. Integrity constraints can be made with arbitrary predicates.
but it may be too costly to test.
This is why most DBMSs only allow tests with minimal overhead.

i) NOT NULL Constraint.

ii) UNIQUE $(A_1, A_2, \cdots, A_n)$ Constraint

Attributes $A_1, A_2, \cdots, A_n$ forms a candidate key.

Remark Candidate key attributes are permitted to be NULL.

iii) CHECK (predicate)

Example. check (season in ('spring', 'summer', 'Fall', 'Winter')).

Remark. None of the wildly used DBMSs allows predicates containing subquery.

iv) Referential-integrity Constraints (Subset dependencies)

for relations r_1, r_2, let set of attributes of r_1 as R_1, r_2 as R_2,

with Candidate keys C_1, C_2 respectively.

Note that C_1, C_2, R_1, R_2 are all sets, which can be multiple attributes.

We say $\alpha \subset R_2$ is a foreign key referencing C_1 in r_1

if for all tuple $t_2 \in r_2$, there exists tuple $t_1 \in r_1$ such that $t_2.\alpha = t_1.C_1$.

Remark. t_1 cannot be deleted until every tuple $t \in r_2$ where $t.\alpha = t_1.C_1$ is deleted.

This can be made easier with ON DELETE/UPDATE CASCADE.

v) Integrity constraint violation during a transaction.

INITIALLY DEFERRED: This constraint must be checked at the end of transaction.

DEFERRABLE: This constraint can be deferred when desired.

vi) ASSERTION

A predicate expressing a condition that the database has to follow.

It is the most general form of constraints.

Remark This adds significant overhead, so it has to be made with care.

(4.5) SQL Data Types and Schema.

i) Date and Time types.

DATE (year, month, day), TIME(hour, minute, second), TIMESTAMP(DATE+TIME)

EXTRACT (any component of time), TIMEZONE_HOUR/MINUTE

CURRENT_TIME/DATE/TIMESTAMP, LOCALTIME/TIMESTAMP.

INTERVAL (subtraction between DATE/TIME/TIMESTAMP).

ii) CREATE INDEX

Def. An index on an attribute or a relation
is a data structure that allows DBMS
to find tuples for that attribute efficiently, i.e. without full scan.

iii) Large object data types.

CLOB (Character data), BLOB(Binary data)

iv) User-defined types

	Strongly typed?	Constraints?	Defaults?
CREATE TYPE	O: needs CAST. easy to detect mistakes.	X:	X
CREATE DOMAIN	X: no problem if underlying type is same.	O: any predicate can be used. ex) only specified values. NOT NULL.	O

v) CREATE TABLE LIKE - WITH DATA

Creation of table with the existing schema and existing rows.

vi) CATALOG ∋ SCHEMA ∋ TABLE or VIEWS

Remark With higher level hierarchy, it is easier to work with multiple apps, or multiple versions of apps, as it solves name clashing problem.

4.6 Authorization.

GRANT / REVOKE : confer / revoke an authorization

SELECT / INSERT / UPDATE / DELETE : types of authorization (Privilege)

ON relation / specific attributes of a relation

TO user / role

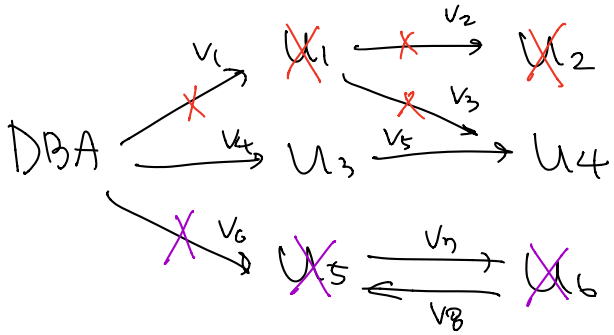
Remark. role-based authorization.

Remark. Functions or Procedures run with the privilege of the user that created it, but with SQL SECURITY INVOKER, it can be run with the privilege in which the user that invoked it.

Remark. Ultimate form of authorization is given to the database admin.

Def. User name PUBLIC refers to all the current and future users.

* Authorization Graph.



User has authorization if and only if there is a path from user to DBA.

Remark. GRANTED BY is used to specify who granted whom in what specific role.

Example 1. Cascading revocation

If u_1 is revoked (v_1 cut), then v_2, v_3 automatically cut.

u_2 is revoked, but u_4 is not because u_3 also gave privilage.

Remark. RESTRICT can prevent this.

Example 2 A pair of devious users.

If u_5 is revoked (v_6 cut), u_5 and u_6 doesn't have any vertex to DBA.

So both are revoked.