Structured Query Language

Год	Название	Иное название	Изменения
1986	SQL-86	SQL-87	Первый вариант стандарта, принятый институтом ANSI и одобренный ISO в 1987 году.
1989	SQL-89	FIPS 127-1	Немного доработанный вариант предыдущего стандарта.
1992	SQL-92	SQL2, FIPS 127-2	Значительные изменения (ISO 9075); уровень *Entry Level* стандарта SQL-92 был принят как стандарт FIPS 127-2.
1999	SQL:1999	SQL3	Добавлена поддержка регулярных выражений, рекурсивных запросов, поддержка триггеров, базовые процедурные расширения, нескалярные типы данных и некоторые объектно-ориентированные возможности.
2003	SQL:2003		Введены расширения для работы с XML-данными, оконные функции (применяемые для работы с OLAP-базами данных), генераторы последовательностей и основанные на них типы данных.
2006	SQL:2006		Функциональность работы с XML-данными значительно расширена. Появилась возможность совместно использовать в запросах SQL и XQuery.
2008	SQL:2008		Улучшены возможности оконных функций, устранены некоторые неоднозначности стандарта SQL:2003
2011	SQL:2011		Добавленные временные типы данных (PERIOD FOR).Улучшения для «оконных» функций и условия FETCH.
2016	SQL:2016		Добавлены построчный pattern matching (проверка на соответствие шаблону), полиморфные табличные функции, JSON.

Типы данных в разных СУБД

Data type	**Access**	**SQLServer**	**Oracle**	**MySQL**	**PostgreSQL**
boolean	Yes/No	Bit	Byte	N/A	Boolean
integer	Number (integer)	Int	Number	Int Integer	Int Integer
float	Number (single)	Float Real	Number	Float	Numeric
currency	Currency	Money	N/A	N/A	Money
string (fixed)	N/A	Char	Char	Char	Char
string (variable)	Text (<256) Memo (65k+)	Varchar	Varchar Varchar2	Varchar	Varchar
binary object	OLE Object Memo	Binary (fixed up to 8K) Varbinary (<8K) Image (<2GB)	Long Raw	Blob Text	Binary Varbinary

Подмножества команд SQL

операторы манипуляции данными (Data Manipulation Language, **DML**):

- **SELECT** считывает данные, удовлетворяющие заданным условиям;
- **INSERT** добавляет новые данные;
- **UPDATE** изменяет существующие данные;
- **DELETE** удаляет данные;

операторы определения данных (Data Definition Language, **DDL**):

- **CREATE** создает объект БД (саму базу, таблицу, представление, пользователя и т. д.);
- **ALTER** изменяет объект;
- **DROP** удаляет объект;

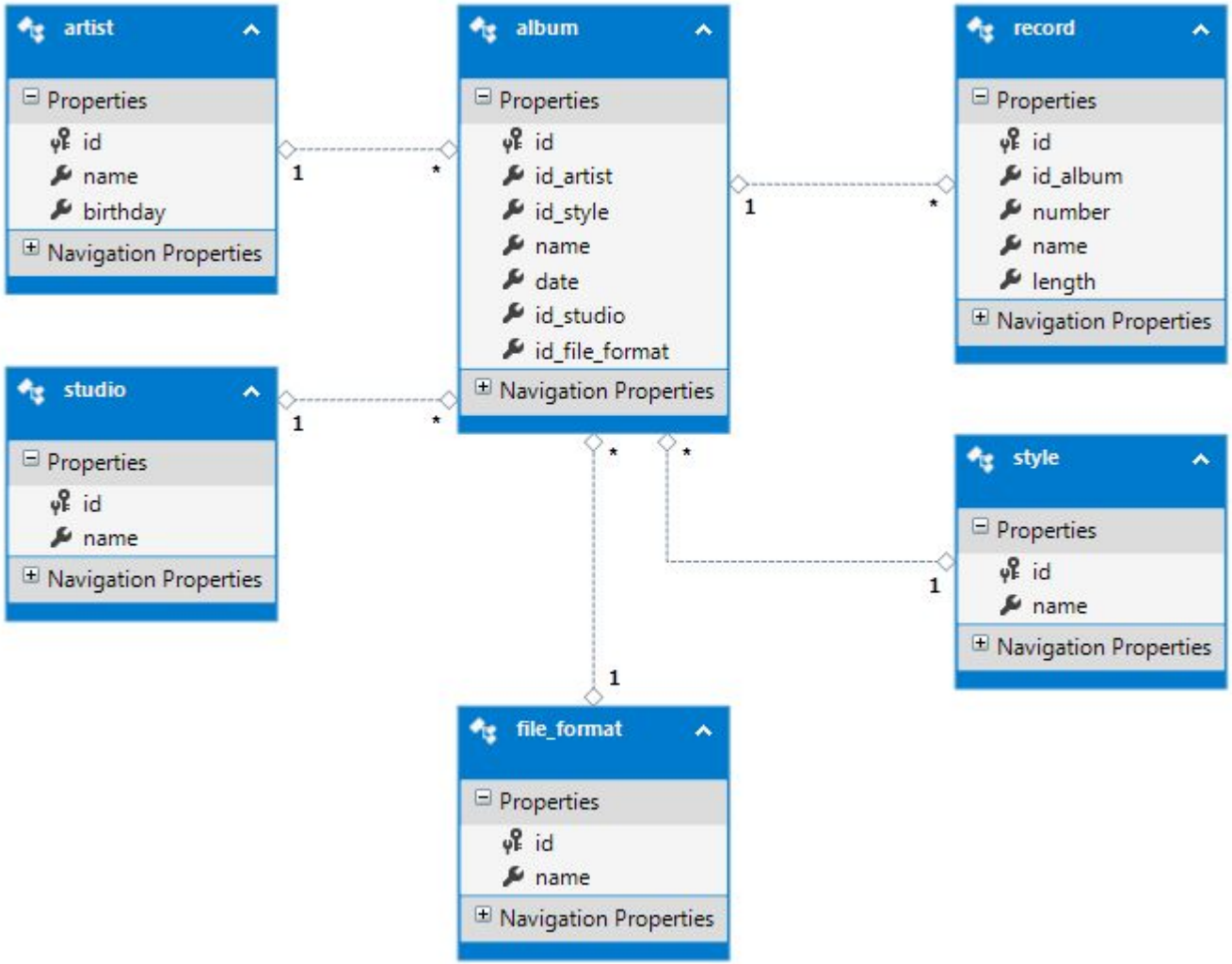
операторы управления транзакциями (Transaction Control Language, **TCL**):

- **COMMIT** применяет транзакцию,
- **ROLLBACK** откатывает все изменения, сделанные в контексте текущей транзакции,
- **SAVEPOINT** делит транзакцию на более мелкие участки.

операторы определения доступа к данным (Data Control Language, **DCL**):

- **GRANT** предоставляет пользователю (группе) разрешения на определенные операции с объектом;
- **REVOKE** отзывает ранее выданные разрешения;
- **DENY** задает запрет, имеющий приоритет над разрешением;

БД для примеров



SELECT

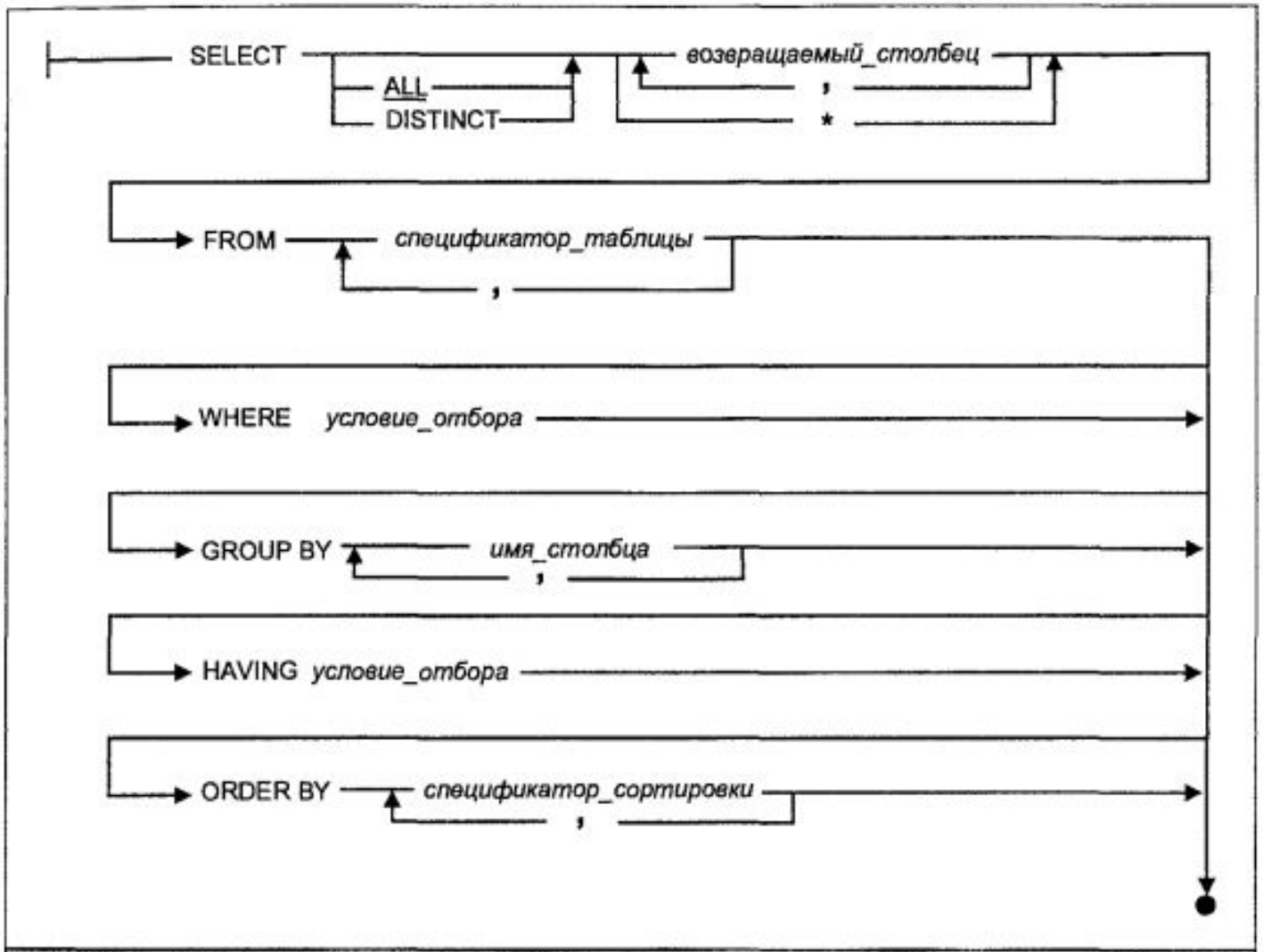
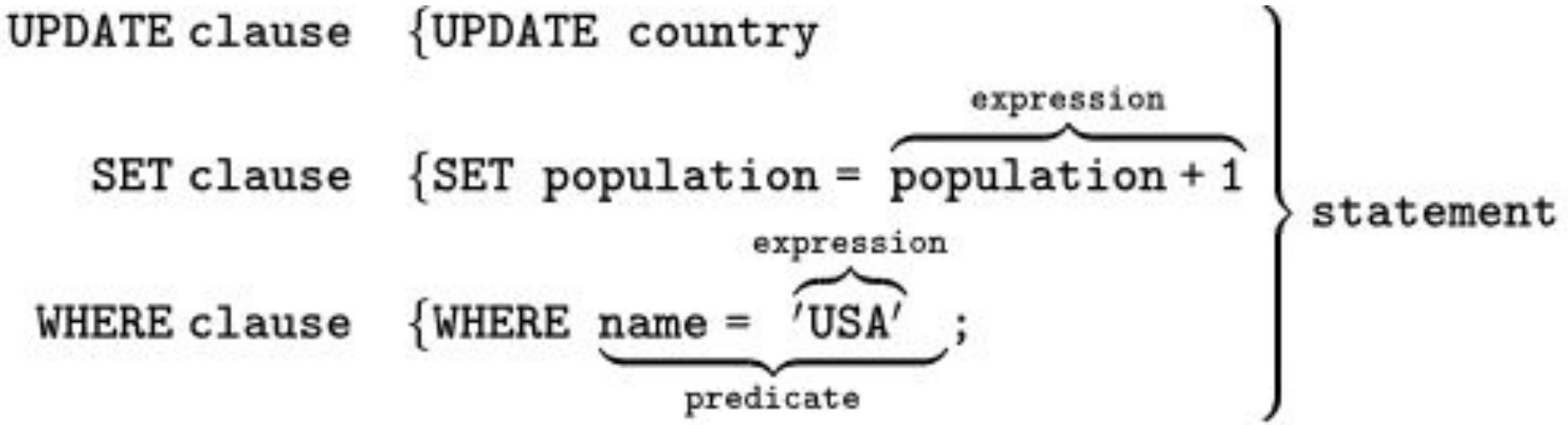


Рис. 6.1. Синтаксическая диаграмма инструкции SELECT


UPDATE clause
SET clause
WHERE clause
UPDATE country
SET population = population + 1
WHERE name = 'USA' ;
expression
expression
predicate
statement

SELECT

```
SELECT *
    FROM artists;

SELECT albums.name, albums.id, albums.date
    FROM albums;

SELECT name, length
    FROM records
    WHERE records.id = 2;
```

Условия в SQL

- AND, OR, NOT
- >, <, <=, >=, =, !=, <>
- IS NULL, IS NOT NULL

Сложные условия в SQL

- LIKE
- BETWEEN
- IN, NOT IN
- ANY, ALL
- EXISTS
- CASE … WHEN … THEN … ELSE … END

Экзотические условия в SQL

- COALESCE
- NULLIF
- …

Примеры с несколькими условиями

```
SELECT albums.id < 8 , albums.date
FROM albums
WHERE albums.id = 1 OR
(albums.name NOT LIKE '%th%' AND
albums.date BETWEEN '1990-01-01' AND '2000-12-31' AND
albums.date NOT BETWEEN '1997-01-01' AND '1998-12-31');

SELECT artists.birthday, artists.name
FROM artists
WHERE artists.name != 'ABBA' AND
(artists.birthday IS NOT NULL OR id IN (4, 8, 15, 16, 413));
```

Фильтрация результатов и псевдонимы

```
SELECT DISTINCT length FROM records;

SELECT DISTINCT a.id_studio FROM albums a;

SELECT a.id_studio,
    CASE
        WHEN a.date < '1950-01-01' THEN 'ancient'
        WHEN a.date <= '1990-01-01' THEN 'nope'
        WHEN a.date > '1990-01-01' THEN 'yep'
        ELSE 'WAT' END AS result,
    a.name
FROM albums a
WHERE a.id < 100 AND a.id_file_format = 1;
```

Троичная логика

Таблица 6.1. Таблица истинности оператора AND

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Таблица 6.2. Таблица истинности оператора OR

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Таблица 6.3 Таблица истинности оператора NOT

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

1. p AND q		p		
		True	**False**	**Unknown**
q	**True**	True	False	Unknown
	False	False	False	False
	Unknown	Unknown	False	Unknown

p OR q		p		
		True	**False**	**Unknown**
q	**True**	True	True	True
	False	True	False	Unknown
	Unknown	True	Unknown	Unknown

q	NOT q
True	False
False	True
Unknown	Unknown

p = q		p		
		True	**False**	**Unknown**
q	**True**	True	False	Unknown
	False	False	True	Unknown
	Unknown	Unknown	Unknown	Unknown

Схемы сложных условий

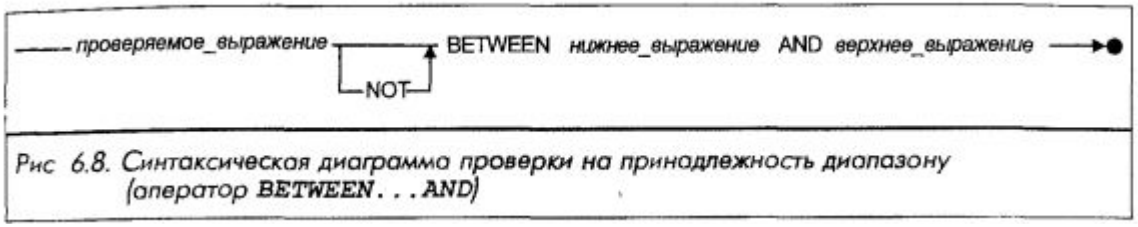


Рис 6.8. Синтаксическая диаграмма проверки на принадлежность диапазону (оператор **BETWEEN. . . AND**)

Рис 6.10 Синтаксическая диаграмма проверки на соответствие шаблону (оператор **LIKE**)

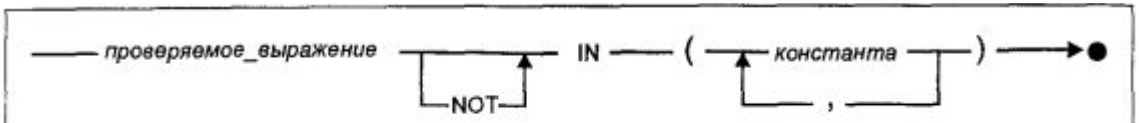


Рис 6.9 Синтаксическая диаграмма проверки на членства в множестве (оператор **IN**)

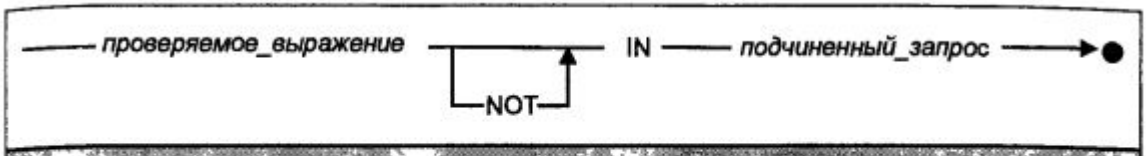


Рис. 9.4. Синтаксическая диаграмма проверки на принадлежность результатам подчиненного запроса (предикат IN)

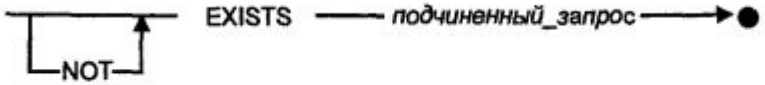


Рис. 9.5. Синтаксическая диаграмма проверки на существование (предикат EXISTS)

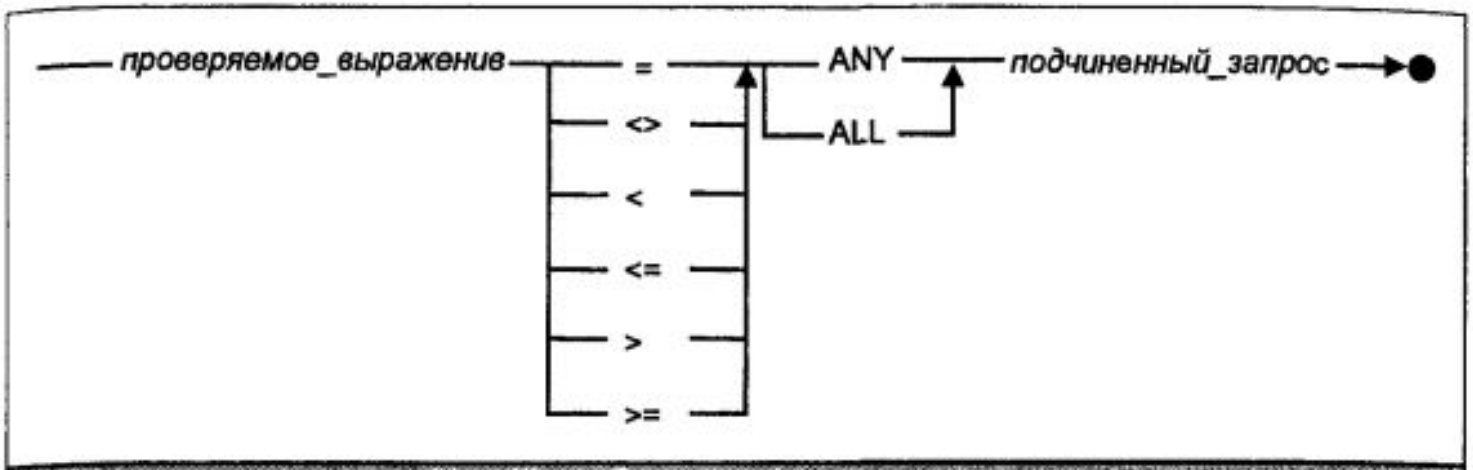


Рис. 9.6. Синтаксическая диаграмма многократного сравнения (предикаты ANY и ALL)

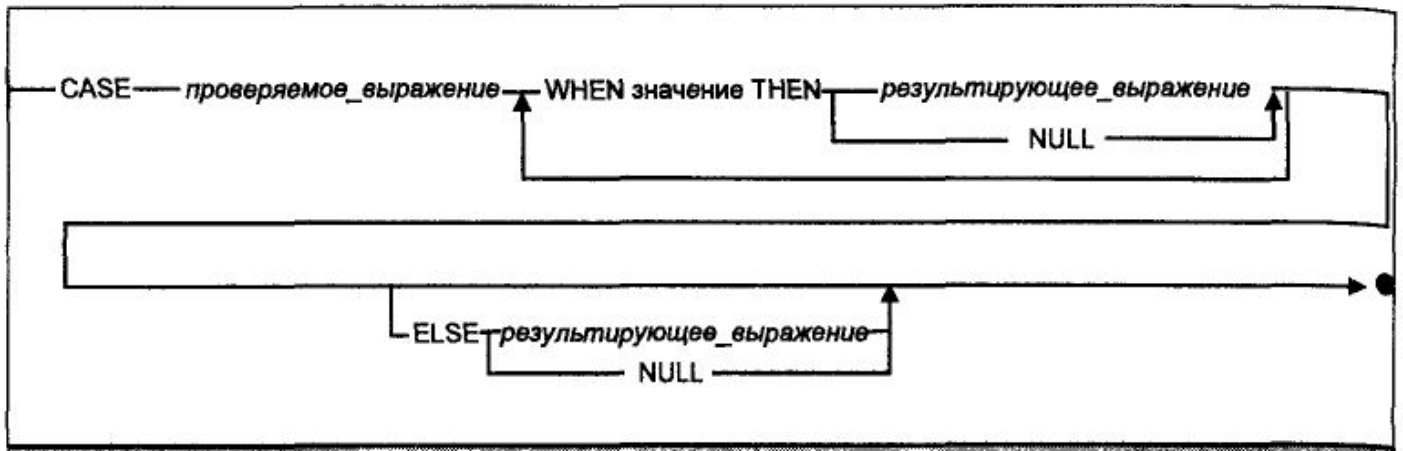


Рис. 9.10. Альтернативный синтаксис оператора CASE (SQL2)

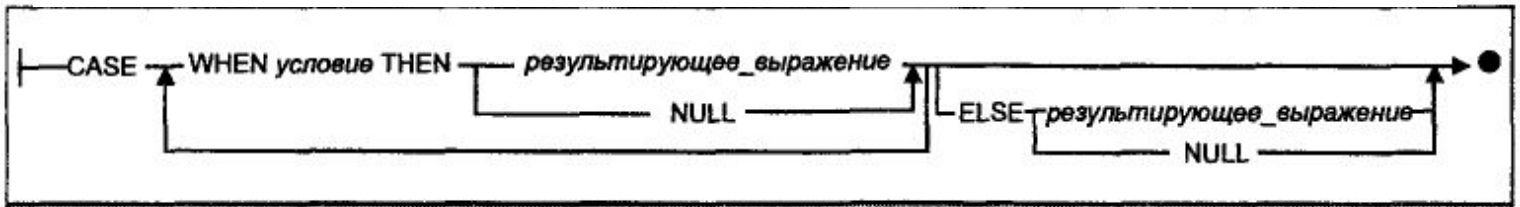


Рис. 9.8. Синтаксическая диаграмма оператора CAST (SQL2)

Сортировка выборки

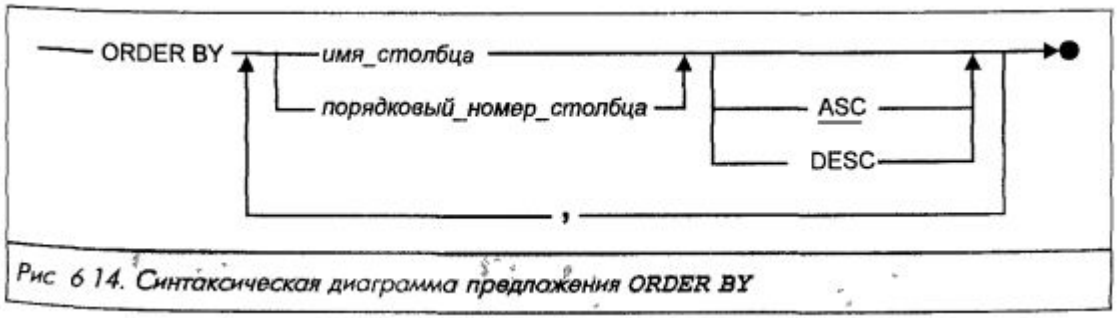


Рис 6 14. Синтаксическая диаграмма предложения ORDER BY

```
SELECT *
FROM records r
WHERE r.id_album = 1
ORDER BY r.length;

SELECT DISTINCT r.number
FROM records r
ORDER BY r.number DESC;
```

Агрегатные функции

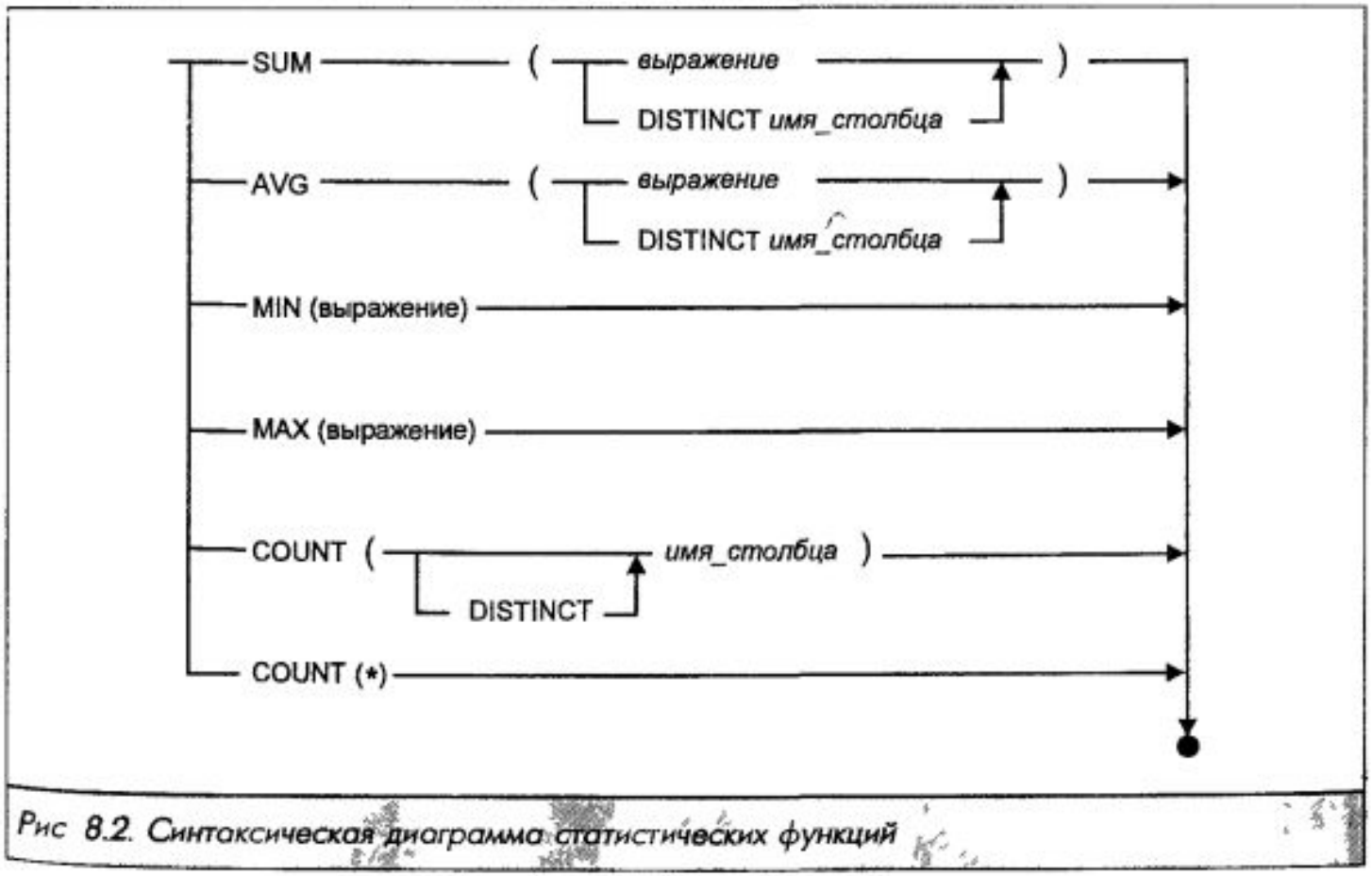


Рис 8.2. Синтаксическая диаграмма статистических функций

Агрегатные функции

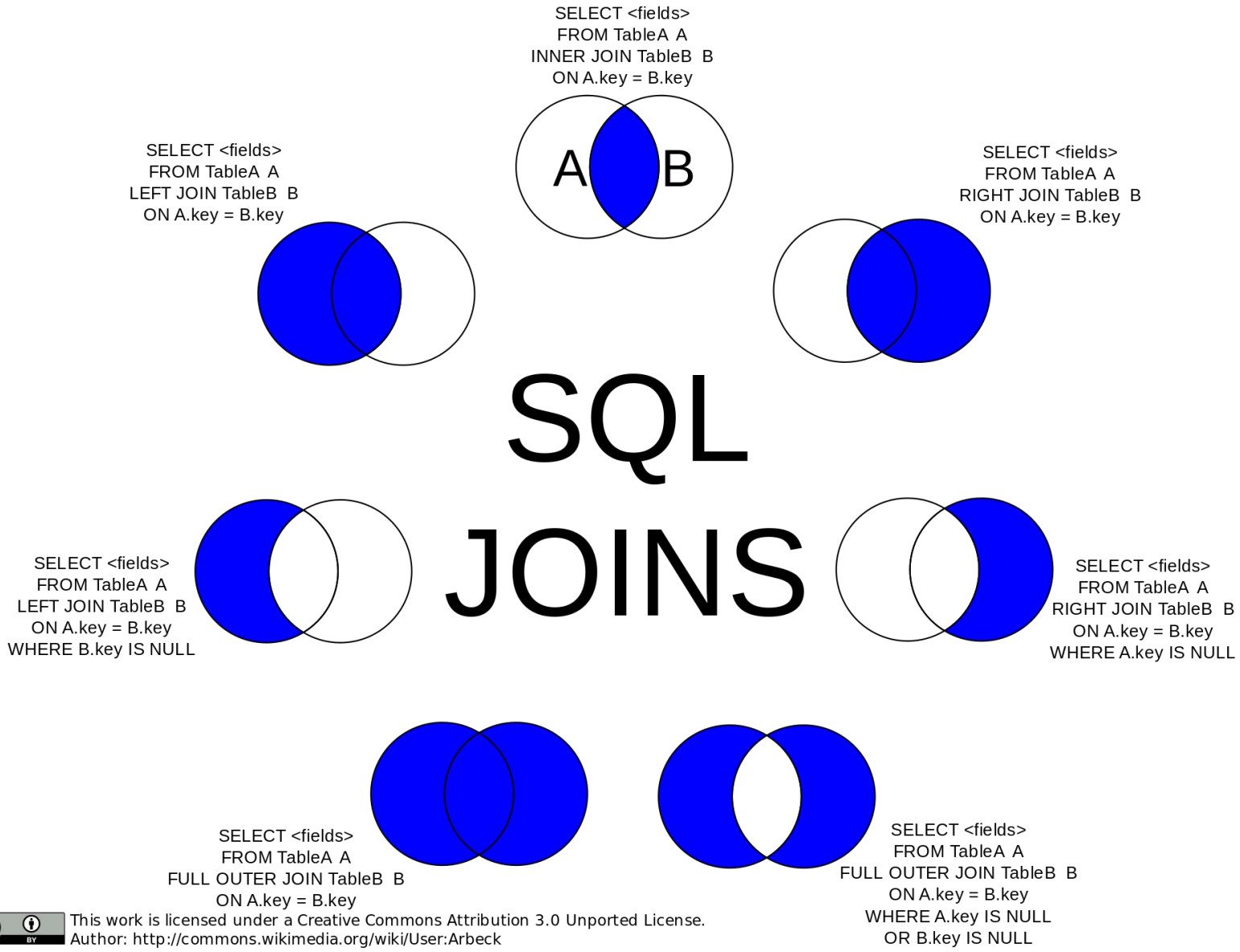
```
SELECT count(*)
FROM records r
WHERE r.length < 5;

SELECT count(DISTINCT r.length)
FROM records r ;

SELECT MAX(a.birthday) max_date, MIN(a.birthday) min
FROM artists a;

SELECT AVG(age(a.date))
FROM albums a
WHERE a.name LIKE '%The%';
```

Группировка

```
SELECT a.id_artist "artist id", AVG(age(a.date)) AS "average album age"
FROM albums a
WHERE a.id_file_format = 1
GROUP BY a.id_artist;
```

```
SELECT AVG(length(r.name)) "average name length", MAX(r.length) "max track length"
FROM records r
GROUP BY r.id_album;
```

Горизонтальное соединение результатов запроса



Объединение результатов запроса

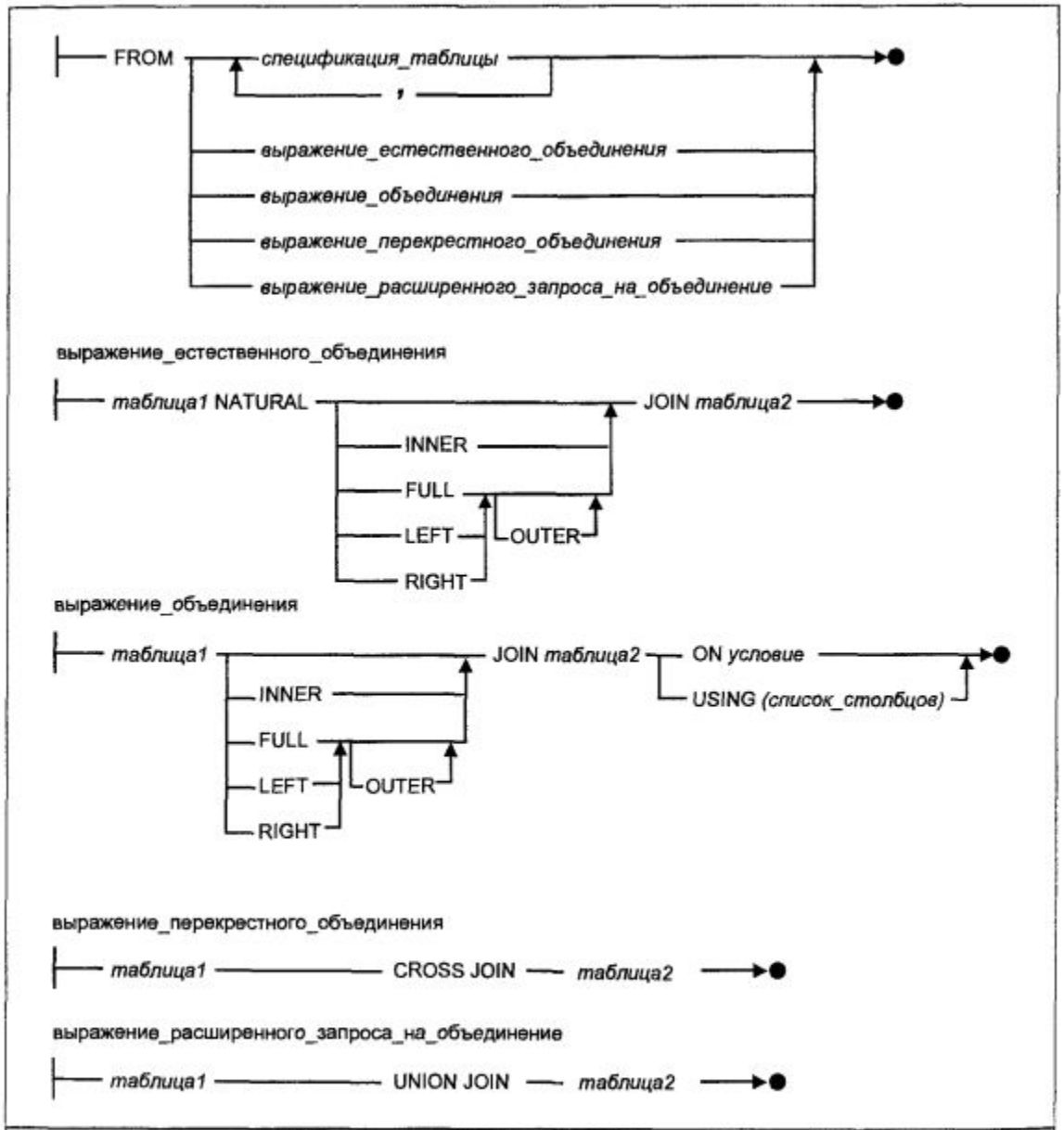


Рис 7 13 Расширенное предложение **FROM** в стандарте SQL2

Примеры объединения таблиц

```
SELECT *
FROM albums a
    INNER JOIN records r
    ON a.id = r.id_album;

SELECT r.name, a.date, ar.birthday, f.*
FROM albums a
    LEFT OUTER JOIN artists ar
    ON ar.id = a.id_artist
    LEFT OUTER JOIN studio s
    ON s.id = a.id_studio
    LEFT OUTER JOIN file_format f
    ON f.id = a.id_file_format
    LEFT OUTER JOIN styles st
    ON st.id = a.id_style
    LEFT OUTER JOIN records r
    ON a.id = r.id_album
WHERE ar.birthday BETWEEN '01-10-1970' AND '10-12-1990'
    AND r.length > 4;
```

Вложенные запросы

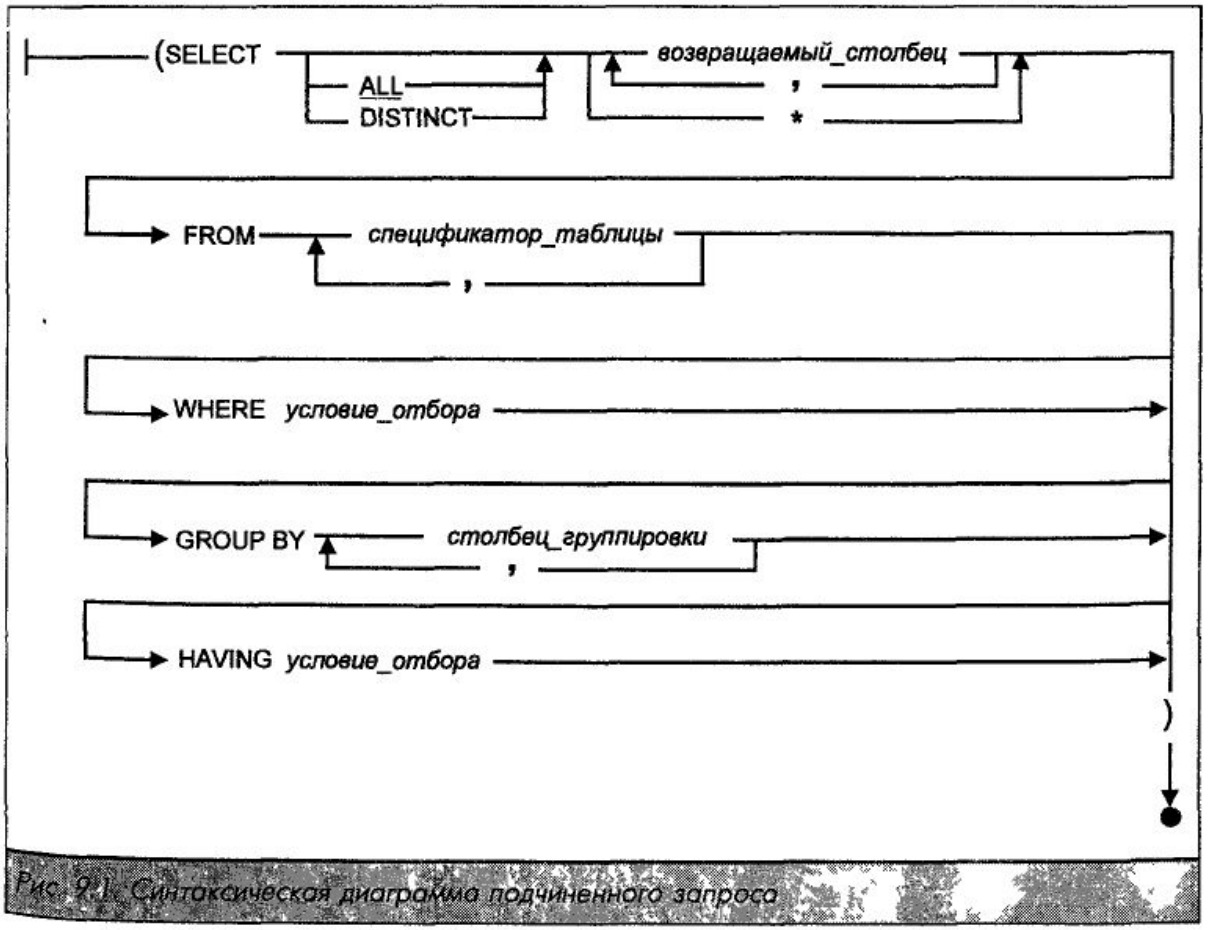


Рис. 9.1. Синтаксическая диаграмма подчиненного запроса

Примеры подзапросов

```
SELECT r.*, a.name
FROM records r
      INNER JOIN albums a
      ON a.id = r.id_album
WHERE a.id_studio IN
     (SELECT s.id
     FROM studio s
     WHERE s.name LIKE '%USA%');


SELECT *
FROM artists ar
WHERE ar.birthday >
(SELECT MIN(a.date)
FROM albums a);
```

Примеры подзапросов

```
SELECT *
FROM albums a
    INNER JOIN
    (SELECT
        AVG(length(r.name)) "average track name length",
MAX(r.length) "max track length",
        r.id_album
    FROM records r
    GROUP BY r.id_album) rc
    ON rc.id_album = a.id
WHERE rc."average track name length" > 12
  OR rc."max track length" = 6;
```

Предикаты ANY, ALL, EXISTS

```
SELECT *
FROM albums a
    INNER JOIN artists ar
    ON ar.id = a.id_artist
WHERE a.id_studio = ANY
    (SELECT s.id
    FROM studio s
    WHERE s.name LIKE '%USA%');

SELECT *
FROM records rc
WHERE rc.length < ALL
    (SELECT AVG(r.length)
    FROM records r
    WHERE r.number < 10
    GROUP BY r.id_album);
```

Примеры HAVING

```
SELECT a.name
FROM
        (SELECT r.id_album
        FROM records r
        GROUP BY r.id_album
        HAVING SUM(r.length) > 20) rc
        INNER JOIN albums a
        ON a.id = rc.id_album;

SELECT s.name
FROM studio s
        INNER JOIN
        (SELECT a.id_studio
        FROM albums a
        GROUP BY a.id_studio
        HAVING MAX(age(a.date)) >
                (SELECT AVG(age(al.date))
                FROM albums al
                WHERE name NOT LIKE '%The%')) alb
        ON alb.id_studio = s.id;
```

Операторы UNION, INTERSECT, EXCEPT

```
SELECT a.name, a.date
FROM albums a
WHERE a.name NOT LIKE '%The%'
UNION
SELECT a.name, a.date
FROM albums a
WHERE a.id_style = 3;

SELECT r.name
FROM records r
UNION ALL
SELECT a.name
FROM artists a;
```

INSERT

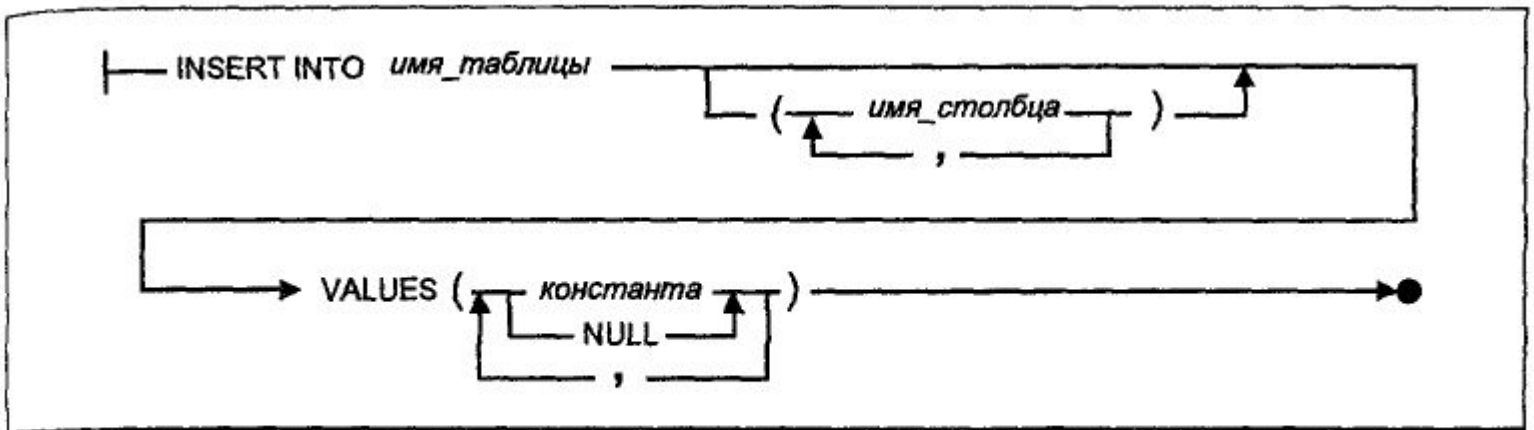


Рис 10.1 Синтаксическая диаграмма однострочной инструкции INSERT

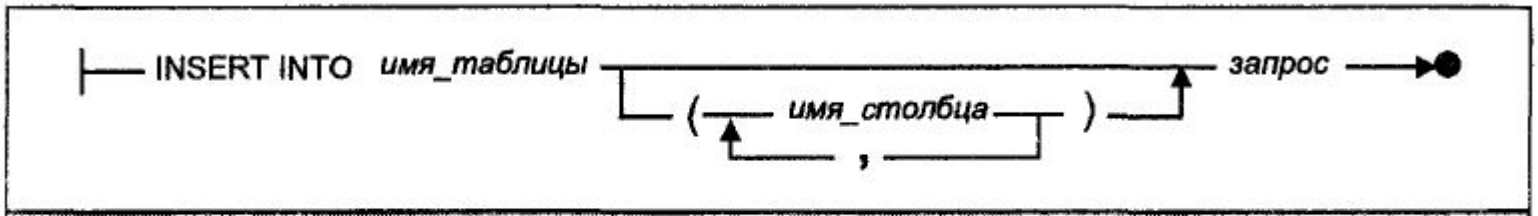


Рис. 10.3. Синтаксическая диаграмма многострочной инструкции INSERT

Примеры INSERT

```
INSERT INTO artists (id, name, birthday) VALUES
(42, 'Nick Cage', '15-03-1900');

INSERT INTO studio (name) VALUES('New studio');

INSERT INTO file_format (id, name) VALUES
(DEFAULT, 'wma');

INSERT INTO styles VALUES (13, '8-bit');
```

UPDATE

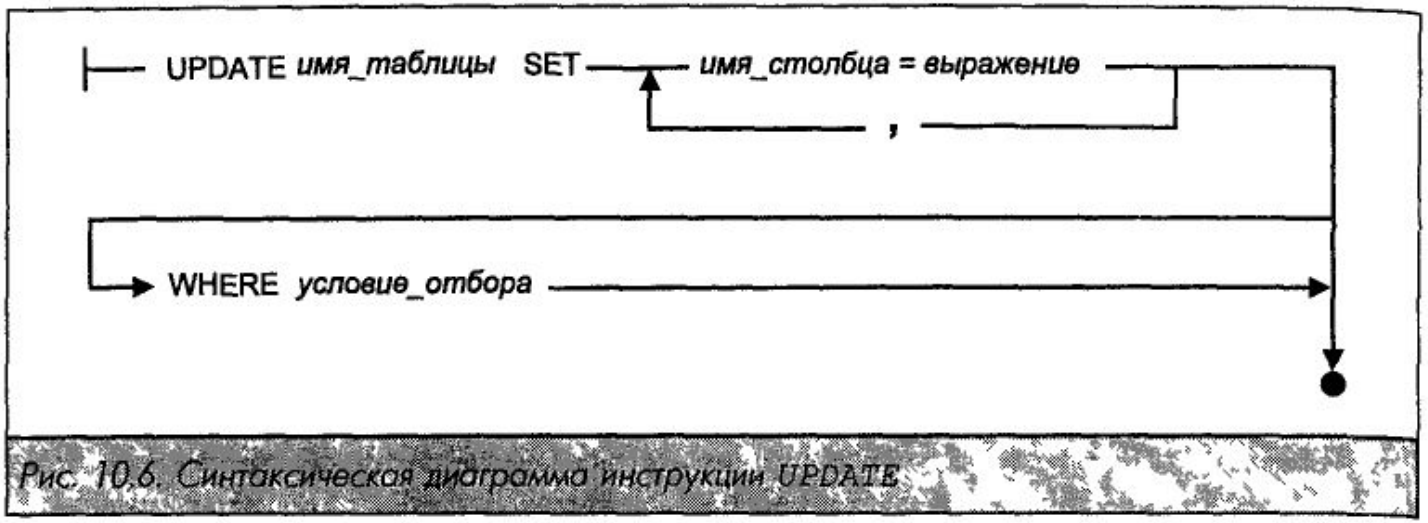


Рис. 10.6. Синтаксическая диаграмма инструкции UPDATE

```
UPDATE artists SET id = 43 WHERE id = 2;

UPDATE records SET length = length * 2
WHERE id_album IN
(SELECT a.id
FROM albums a
WHERE a.id_style=2);
```

DELETE

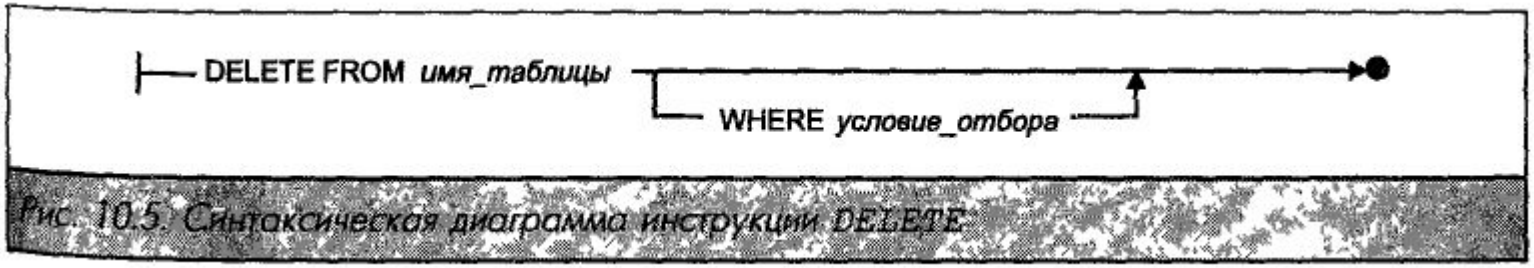


Рис. 10.5. Синтаксическая диаграмма инструкции DELETE

```
DELETE FROM records WHERE records.id < 10;

DELETE FROM albums
WHERE id_studio IN
(SELECT s.id
FROM studio s
WHERE s.id > 5);

DELETE FROM records;
```

Транзакции

- BEGIN TRANSACTION; / BEGIN;
- SAVE TRANSACTION;
- COMMIT TRANSACTION; / COMMIT;
- ROLLBACK TO;
- ROLLBACK;

Объединение запросов с помощью транзакций

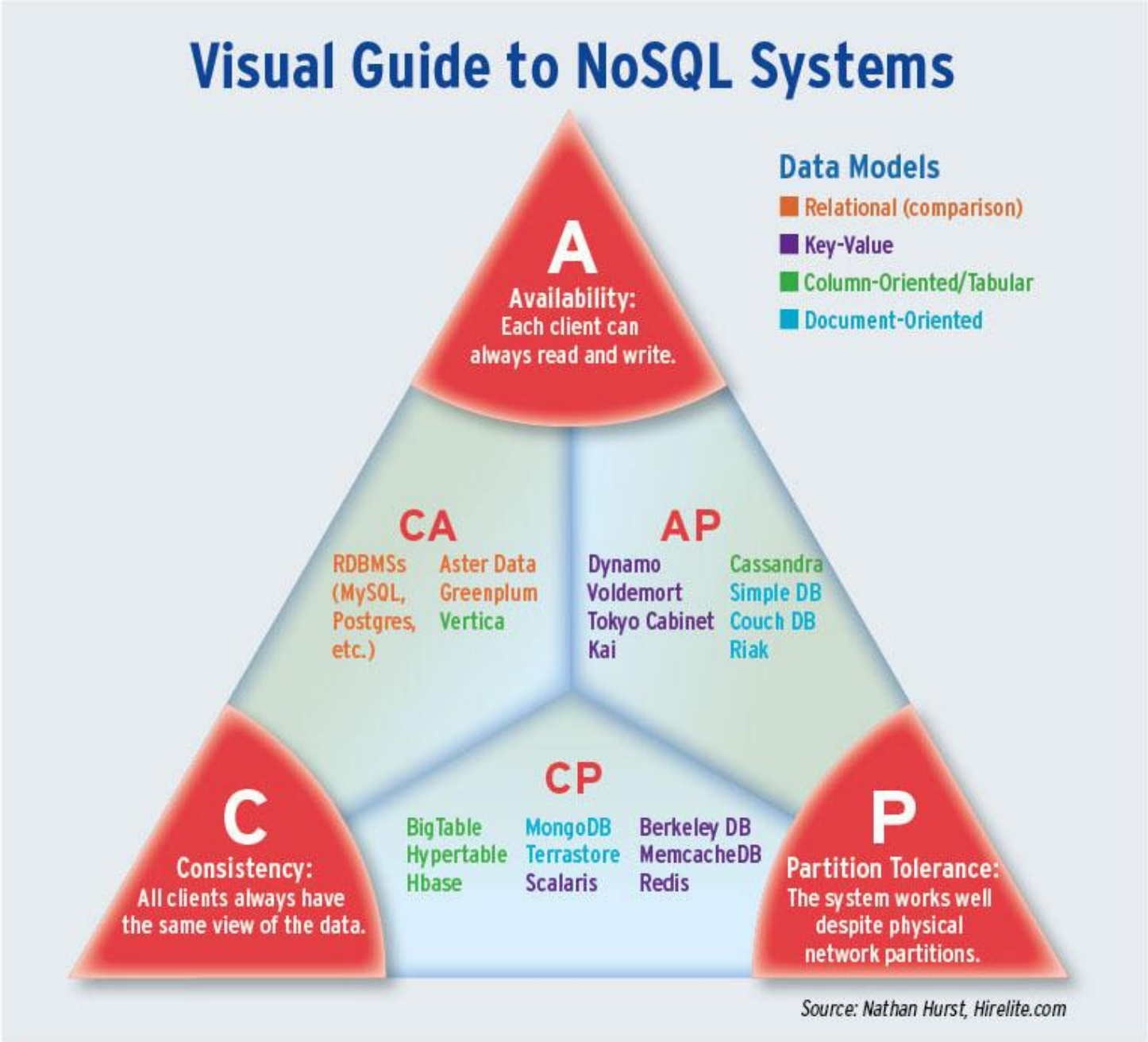
```
BEGIN;

CREATE TABLE studio
(
  id serial NOT NULL,
  name character varying(255) NOT NULL,
  CONSTRAINT pk_studio PRIMARY KEY (id),
  CONSTRAINT uk_studio_name UNIQUE (name)
);

INSERT INTO studio (name) SELECT DISTINCT a.studio FROM albums a;

ALTER TABLE albums ADD COLUMN id_studio integer;

UPDATE albums SET id_studio =
    (SELECT s.id FROM studio s
     WHERE s.name = albums.studio);

ALTER TABLE albums ADD CONSTRAINT fk_album_studio FOREIGN KEY (id_studio) REFERENCES studio (id) ON UPDATE
CASCADE ON DELETE NO ACTION;

ALTER TABLE albums ALTER COLUMN id_studio SET NOT NULL;

ALTER TABLE albums DROP COLUMN studio;

COMMIT;
```

Объединение запросов с помощью транзакций

```
BEGIN;

CREATE TABLE file_format
(
  id serial NOT NULL,
  name character varying(255) NOT NULL,
  CONSTRAINT pk_file_format PRIMARY KEY (id),
  CONSTRAINT uk_file_format_name UNIQUE (name)
);

INSERT INTO file_format (name) SELECT DISTINCT a.format FROM albums a;

ALTER TABLE albums ADD COLUMN id_file_format integer;

UPDATE albums SET id_file_format =
    (SELECT f.id FROM file_format f
     WHERE f.name = albums.format);

ALTER TABLE albums ADD CONSTRAINT fk_album_file_format FOREIGN KEY (id_file_format) REFERENCES file_format (id)
ON UPDATE CASCADE ON DELETE NO ACTION;

ALTER TABLE albums ALTER COLUMN id_file_format SET NOT NULL;

ALTER TABLE albums DROP COLUMN format;

COMMIT;
```

SQL-инъекции

NOSQL


Visual Guide to NoSQL Systems
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
A
Availability:
Each client can always read and write.
CA
RDBMSs (MySQL, Postgres, etc.)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
Kai
Cassandra
Simple DB
Couch DB
Riak
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
C
Consistency:
All clients always have the same view of the data.
P
Partition Tolerance:
The system works well despite physical network partitions.
Source: Nathan Hurst, Hirelite.com


That's all Folks!