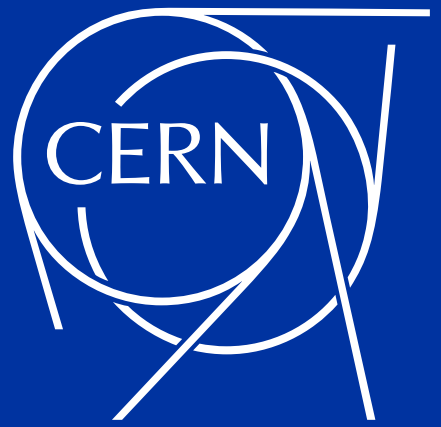


A summary of the most useful new SQL features in 23ai

Andrzej Nowicki

SOUG Day Romandie, Lausanne, 2024

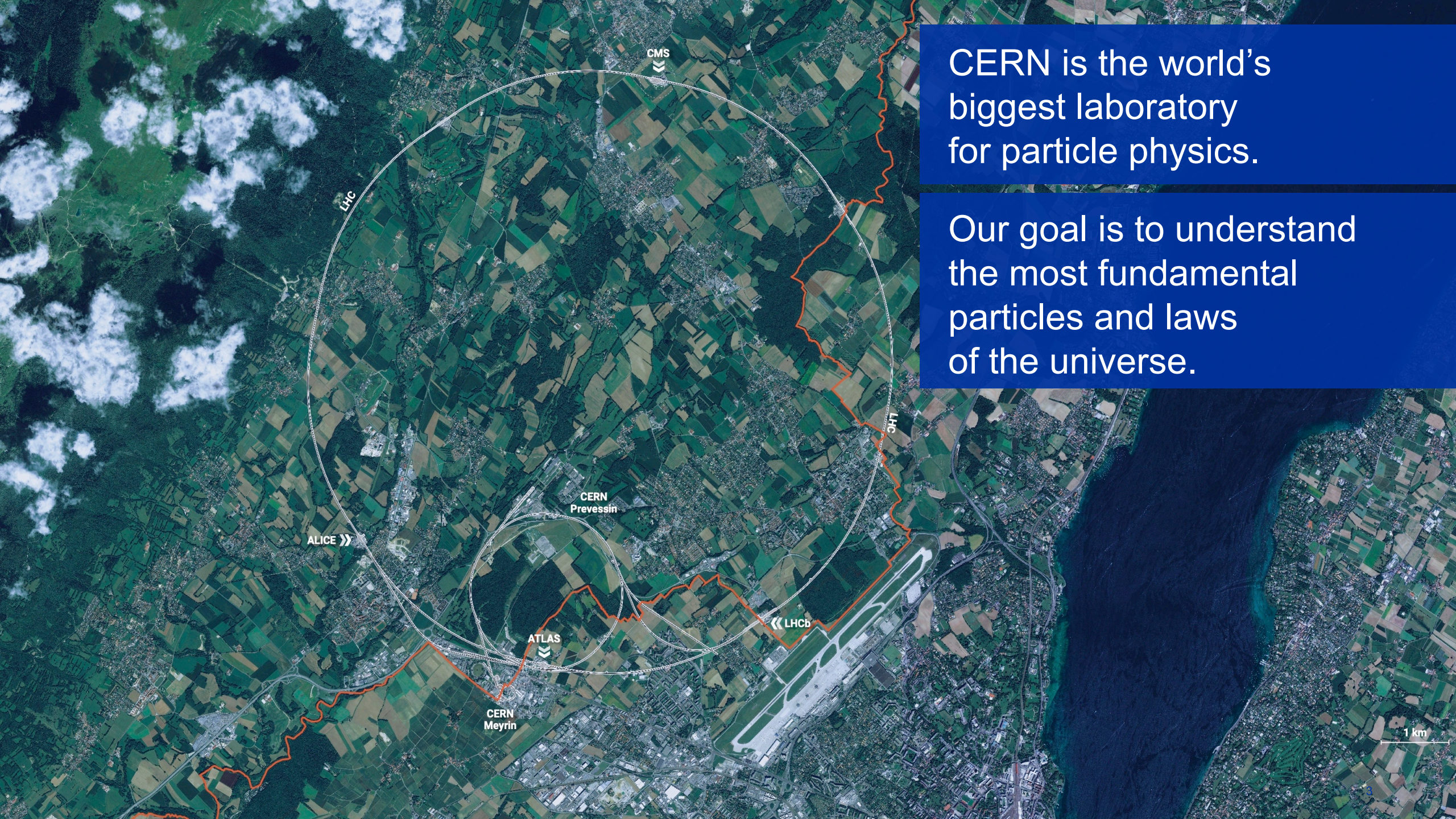
Andrzej Nowicki

12 years of Oracle DB experience
Database Engineer @ CERN since 2020

andrzejnowicki

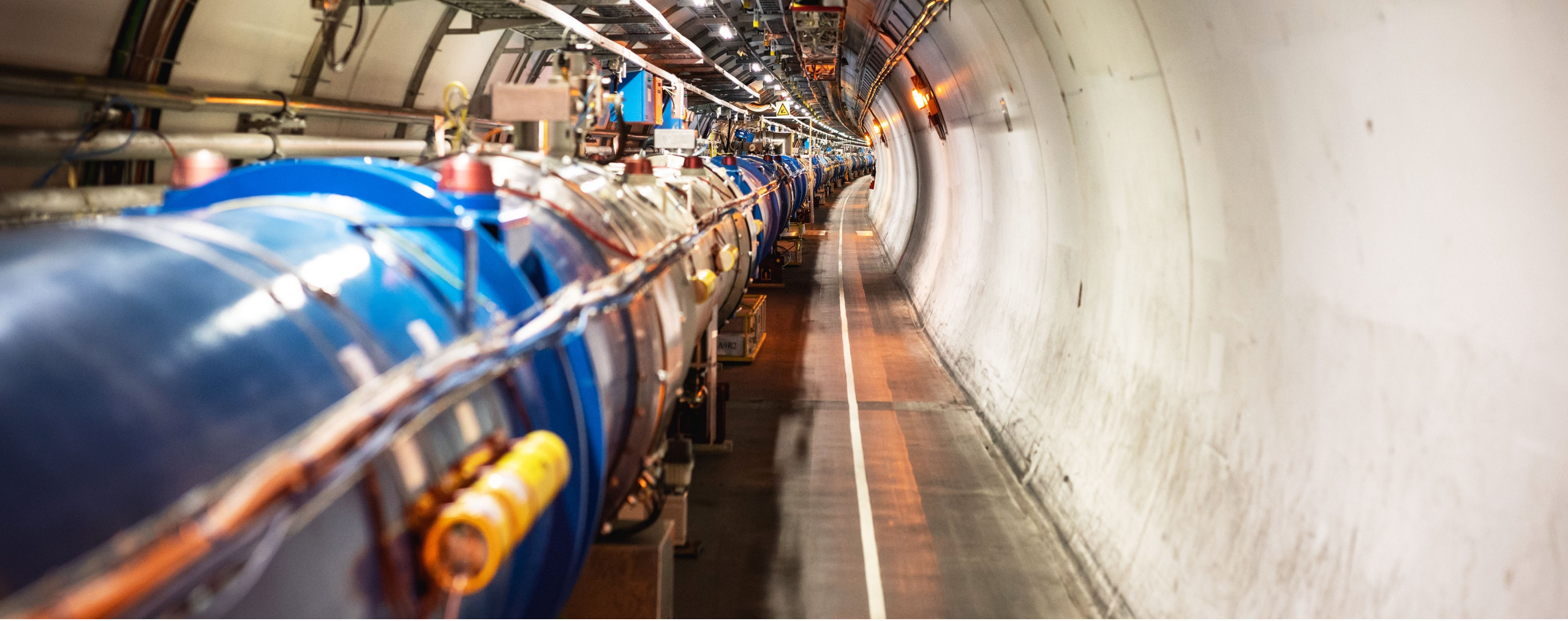
andrzej.nowicki@cern.ch

www.andrzejnowicki.pl

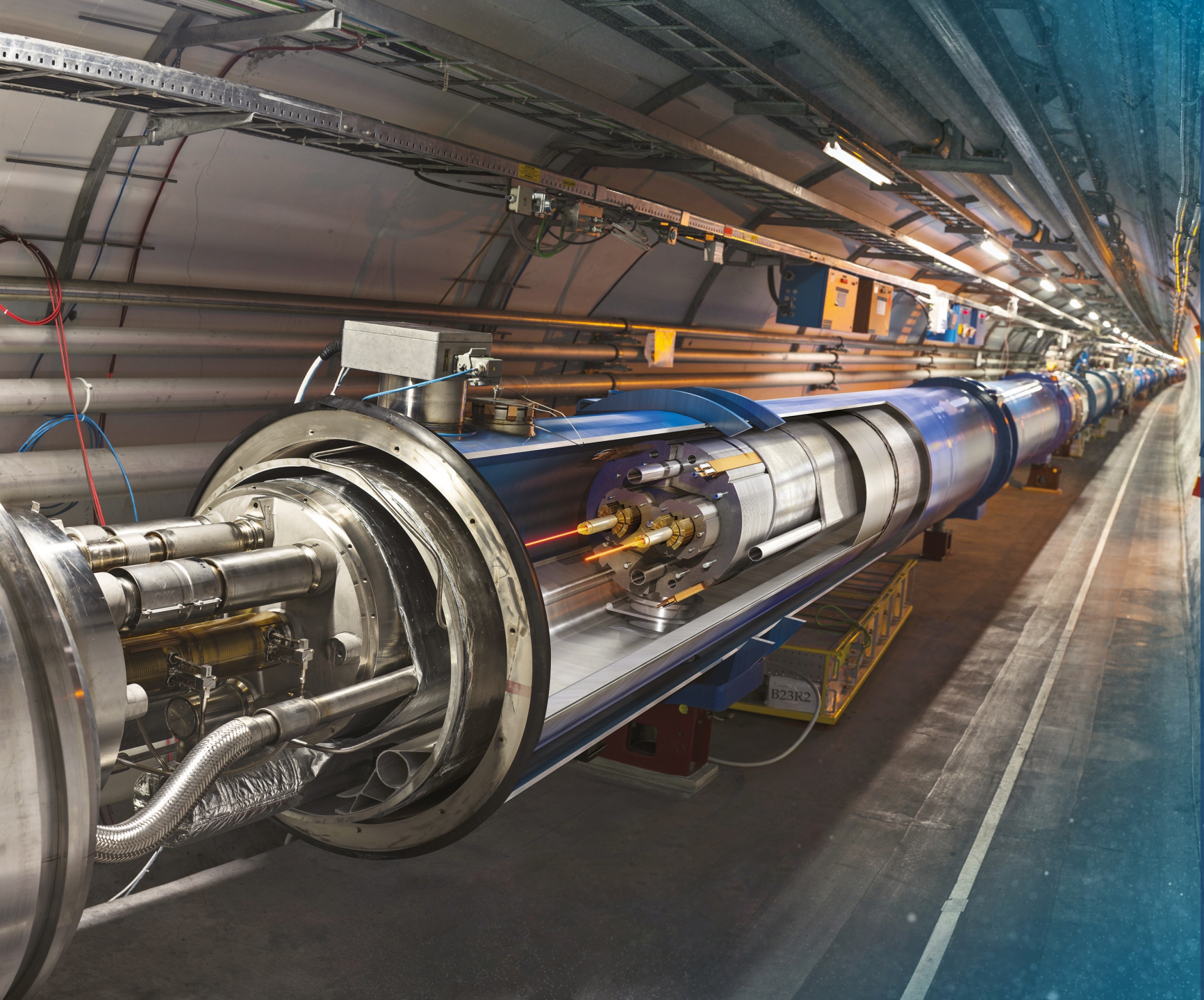


CERN is the world's biggest laboratory for particle physics.

Our goal is to understand the most fundamental particles and laws of the universe.

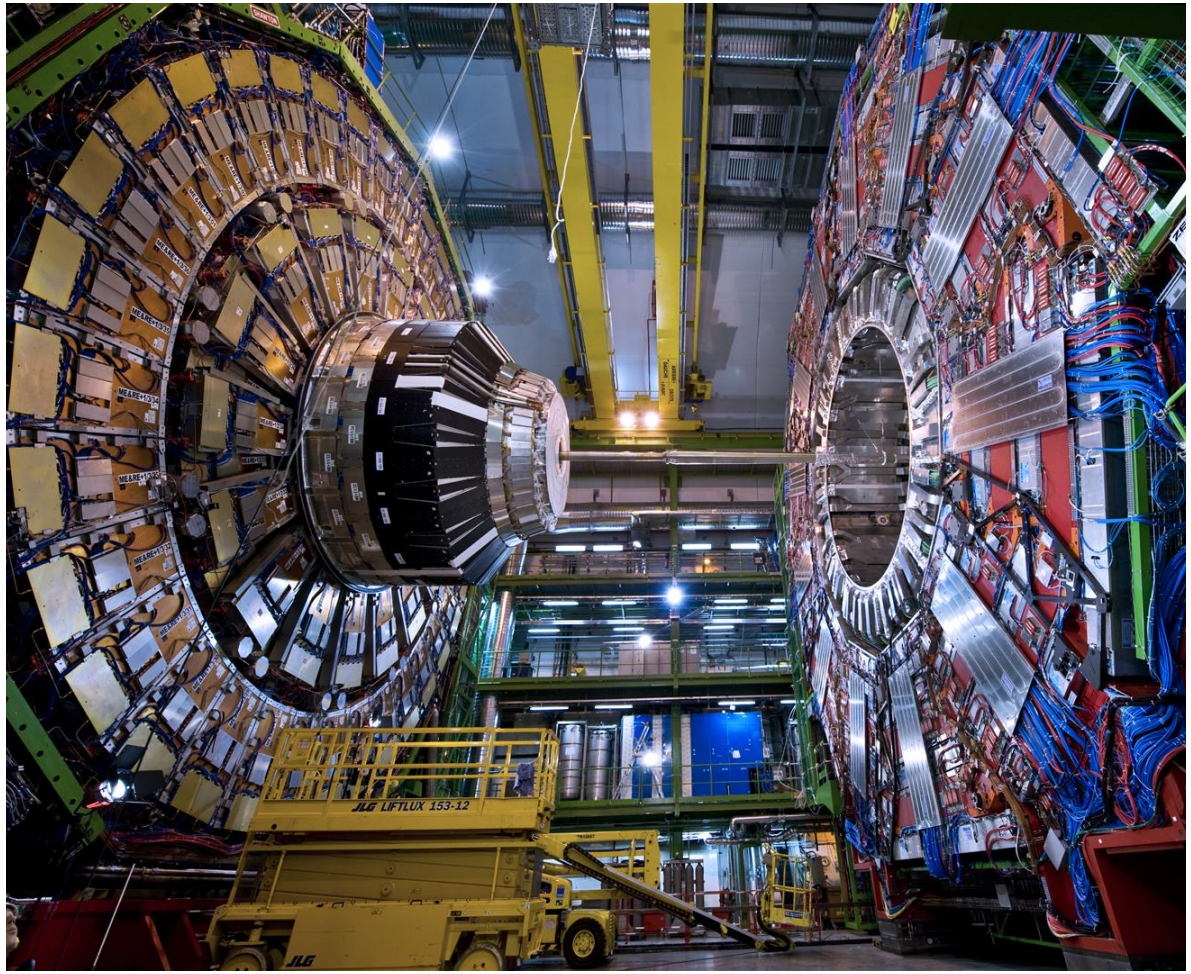


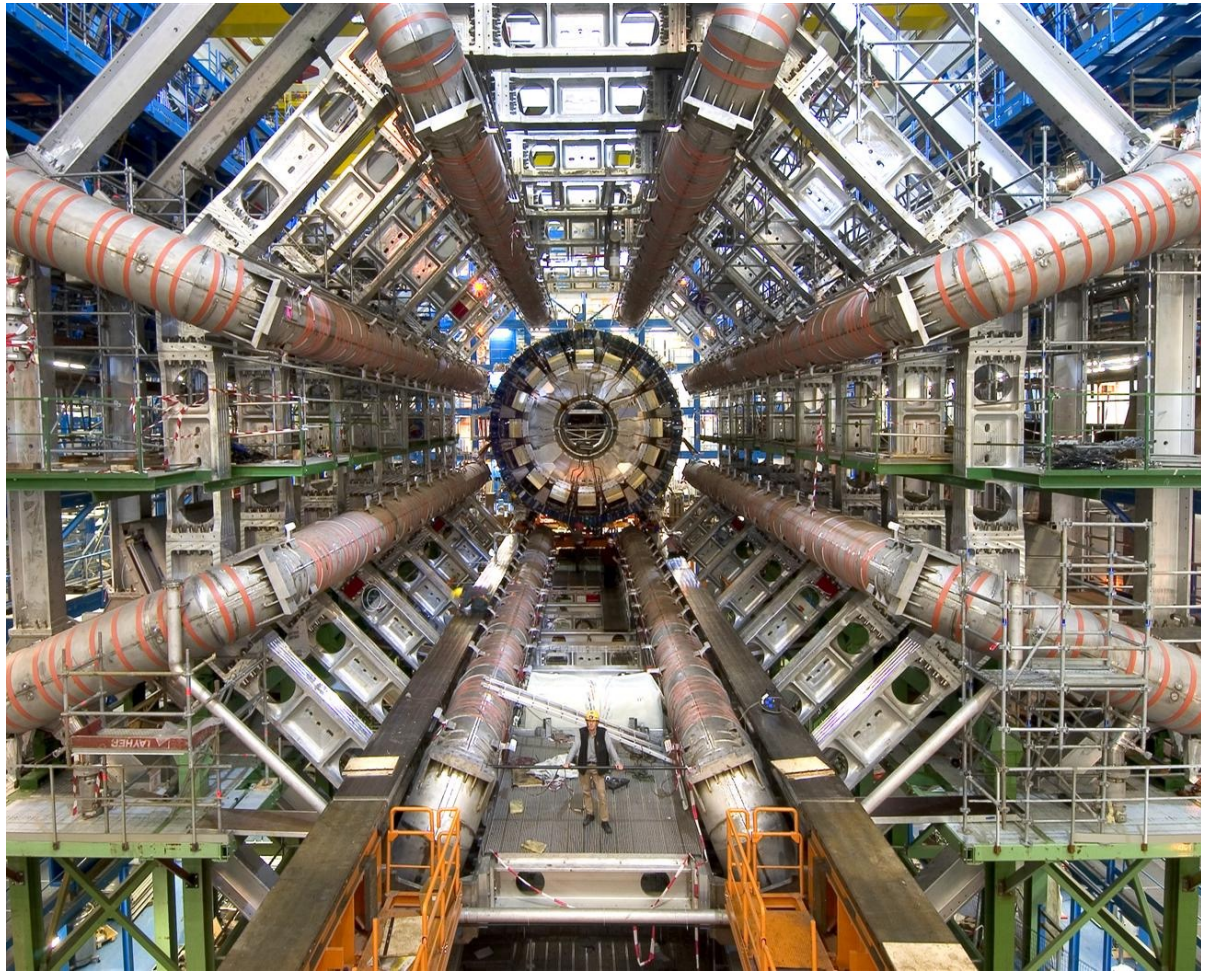
Large Hadron Collider (LHC)



Large Hadron Collider (LHC)

- 27 km (17 mi) in circumference
- About 100 m (300 ft) underground
- Superconducting magnets steer the particles around the ring
- Particles are accelerated to close to the speed of light


JLG LIFTLUX 153-12
JLG



IT @ CERN

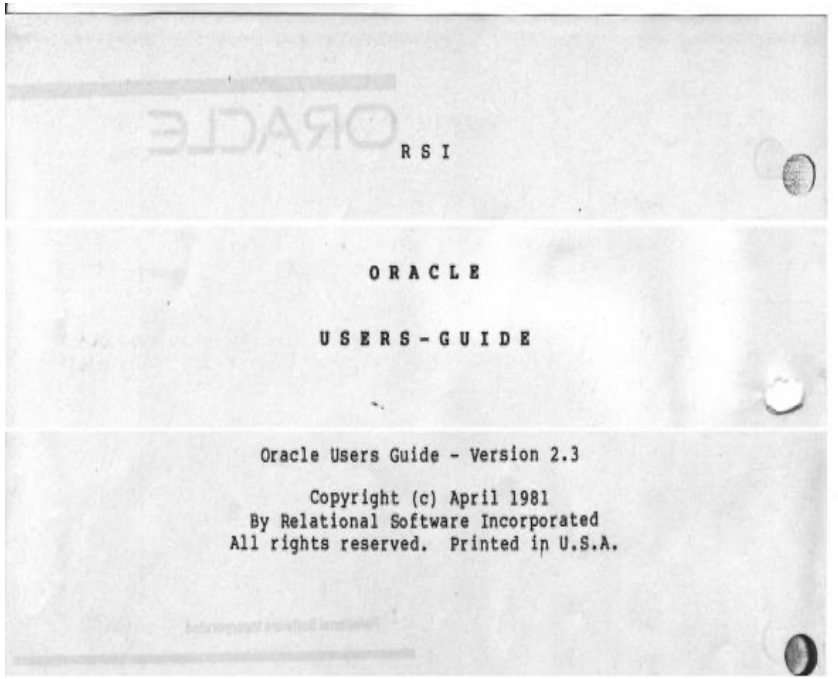
Databases at CERN

Oracle since 1982

- 105 Oracle databases, more than 11.800 Oracle accounts
- RAC, Active Data Guard, GoldenGate, OEM, RMAN, APEX, Cloud…
- Complex environment

Database on Demand (DBoD) since 2011

- ≈600 MySQL, ≈400 PostgreSQL, ≈200 InfluxDB
- Automated backup and recovery services, monitoring, clones, replicas
- HA MySQL clusters (Proxy + primary replica)


RSI

ORACLE

USERS-GUIDE

Oracle Users Guide - Version 2.3

Copyright (c) April 1981
By Relational Software Incorporated
All rights reserved. Printed in U.S.A.

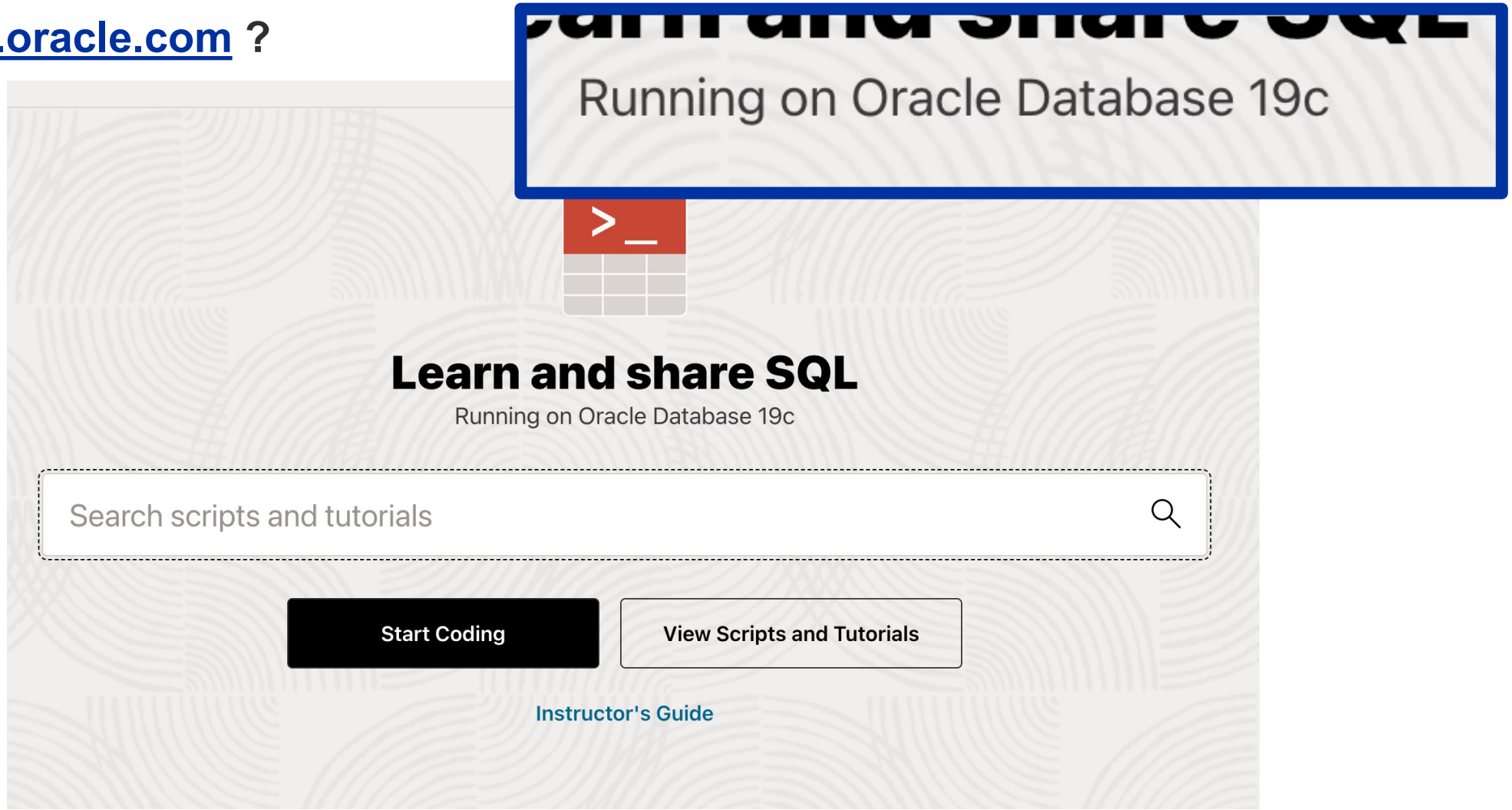
Size of the database environment

	Total size
Oracle	≈5 PB
DBoD (MySQL, PostrgeSQL, InfluxDB)	≈150 TB
Backups	≈3 PB

Oracle 23ai

Where to test 23ai?

https://livesql.oracle.com ?



Where to test 23ai?

Cool, I'll spin up an ADB…

If you are using Autonomous Database with Oracle Database 23ai, the following Oracle Database 23ai functionality is not supported:

Autonomous Database with Oracle Database 23ai in the Paid tier is available in all commercial public cloud regions ***except* the following regions**: South Africa Central: Johannesburg: (JNB), Brazil Southeast: Vinhedo (VCP), Canada Southeast: Toronto (YYZ), and **Switzerland North: Zurich (ZHR)**.

Always Free Autonomous Database with Oracle Database 23ai **is available in the regions**: US West: Phoenix (PHX), US East: Ashburn (IAD), **UK South: London (LHR), France Central: Paris (CDG)**, Australia East: Sydney (SYD), India West: Mumbai (BOM), Singapore (SIN), and Japan East: Tokyo (NRT).

https://docs.oracle.com/en-us/iaas/autonomous-database-serverless/doc/autonomous-always-free-23ai.html

Where to test 23ai?

Oracle Exadata Cloud@Customer
OCI Exadata Database Service
OCI Base Database Service

Oracle Database 23ai Free – https://www.oracle.com/database/free/get-started/

Available as: Docker image, VM VirtualBox, rpm for OEL & RHEL.

Side node, if you have an ARM-based Mac, you can run the docker image.

See Eduardo Claro's and Ron Ekin's blogs:
https://clarodba.wordpress.com/ and https://ronekins.com/

Docker

```
$ colima start --arch x86_64 \
      --vm-type=vz --vz-rosetta --mount-type=virtiofs --memory

$ docker start 23ai

$ sqlplus sys/magicpass@localhost:1623 as sysdba
SQL*Plus: Release 23.0.0.0.0 - Production
Version 23.3.0.23.09

Copyright (c) 1982, 2023, Oracle.  All rights reserved.


Connected to:
Oracle Database 23ai Free Release 23.0.0.0.0 - Develop, Learn, and Run for Free
Version 23.4.0.24.05

SQL>
```

Oracle 23ai
Database New Features

https://docs.oracle.com/en/database/oracle/oracle-database/23/nfcoa/

Oracle Database New Features

Title and Copyright Information
1 Introduction
About
Feature Highlights
2 AI Vector Search
3 Application Development
JSON
SQL
Graph
Microservices
General
Java
JavaScript
Application Connectivity
Database Drivers API Enhancements
4 Data Analytics
5 Data Warehousing/Big Data
6 Cloud Migration
7 Cloud Operations

SQL

Schema Annotations

Schema annotations enable you to store and retrieve metadata about database objects. These are name-value pairs or simply a name. These are free-form text fields applications can use to customize business logic or user interfaces.

Annotations help you use database objects in the same way across all applications. This simplifies development and improves data quality.

View Documentation

Direct Joins for UPDATE and DELETE Statements

Join the target table in `UPDATE` and `DELETE` statements to other tables using the `FROM` clause. These other tables can limit the rows changed or be the source of new values.

Direct joins make it easier to write SQL to change and delete data.

View Documentation

IF [NOT] EXISTS Syntax Support

DDL object creation, modification, and deletion now support the `IF EXISTS` and `IF NOT EXISTS` syntax modifiers. This enables you to control whether an error should be raised if a given object exists or does not exist.

The `IF [NOT] EXISTS` syntax can simplify error handling in scripts and by applications.

View Documentation

New Database Role for Application Developers

Automatic PL/SQL to SQL Transpiler
Client Describe Call Support for Tag Options
DEFAULT ON NULL for UPDATE Statements
DESCRIBE Now Supports Column Annotations
Data Use Case Domain Metadata Support in OCCI
Data Use Case Domains
Error Message Improvement
Extended CASE Controls
GROUP BY Column Alias or Position
Improved TNS Error Messages
Multilingual Engine Support for SQL BOOLEAN Data Type
Oracle C++ Call Interface (OCCI) Support for SQL BOOLEAN Data Type

database must perform.

View Documentation

SQL*Plus Support for SQL BOOLEAN Data Type

SQL*Plus supports the new SQL `BOOLEAN` data type in SQL statements and the `DESCRIBE` command. Enhancements to the `COLUMN` and `VARIABLE` command syntax have also been made.

SQL*Plus scripts can take advantage of the new SQL `BOOLEAN` data type for easy development.

View Documentation

Table Value Constructor

The database's SQL engine now supports a `VALUES` clause for many types of statements. This new clause allows for materializing rows of data on the fly by specifying them using the new syntax without relying on existing tables. Oracle supports the `VALUES` clause for the `SELECT`, `INSERT`, and `MERGE` statements.

The introduction of the new `VALUES` clause allows developers to write less code for ad-hoc SQL commands, leading to better readability with less effort.

View Documentation

Unicode 15.0 Support

The National Language Support (NLS) data files for `AL32UTF8` and `AL16UTF16` character sets are updated to match version 15.0 of the Unicode Standard character database.

This enhancement enables Oracle Database to conform to the latest version of the Unicode Standard.

View Documentation

Previous Page Next Page

Feel free to take photos, but…

The presentation is on my website

https://www.andrzejnowicki.pl/slides/

SELECT without DUAL

```
SQL> select sysdate;

SYSDATE
---------
21-SEP-24

SQL> select 12-2;

      12-2
----------
        10

SQL> select 'Andrzej' FirstName;

FIRSTNA
-------
Andrzej
```

```
SQL> select'Andrzej'FirstName;

FIRSTNA
-------
Andrzej
```

SELECT without DUAL

```
SQL> select0;
select0
*
ERROR at line 1:
ORA-24333: zero iteration count
Help: https://docs.oracle.com/error-help/db/ora-24333/

SQL> select+0;

      +0
----------
       0
```

sqlplus 23ai provides a link to documentation explaining the cause of the error

SELECT without DUAL

```
SQL> selectsysdate;
SP2-0734: unknown command beginning "selectsysd..." - rest of line ignored.
Help: https://docs.oracle.com/error-help/db/sp2-0734/

SQL> select-sysdate;
select-sysdate
       *
ERROR at line 1:
ORA-00932: expression is of data type DATE, which is incompatible with expected
data type NUMBER
Help: https://docs.oracle.com/error-help/db/ora-00932/

SQL> select+sysdate;

+SYSDATE
---------
21-SEP-24
```

SQLPLUS improvements

SET ERRORDETAILS – controls the error message handling URL

ARGUMENT command which is a nicer way to handle arguments like “&1”

New functions:

PING (tnsping other services in the same listener)

OERR

SQLPLUS improvements – oerr – quick demo

```
SQL> oerr ora 25000
Message: "invalid use of bind variable in trigger WHEN clause"
Help: https://docs.oracle.com/error-help/db/ora-25000/
Cause: A bind variable was used in the when clause of a trigger.
Action: Remove the bind variable.  To access the table columns use
       (new/old).column_name.
SQL> oerr ora 600
Message: "internal error code, arguments: [%s], [%s], [%s], [%s], [%s], [%s], [%s], [%s]
Help: https://docs.oracle.com/error-help/db/ora-00600/
Cause: This is the generic internal error number for Oracle program
       exceptions. It indicates that a process has encountered a low-level,
       unexpected condition. The first argument is the internal message
       number. This argument and the database version number are critical in
       identifying the root cause and the potential impact to your system.
SP2-0642: SQL*Plus internal error state 2590, context 52275:32768:0
Help: https://docs.oracle.com/error-help/db/sp2-0642/
Unable to proceed
Disconnected from Oracle Database 23ai Free Release 23.0.0.0.0 - Develop, Learn, and Run
Version 23.4.0.24.05
```

VALUES constructor… the past

```
SQL> create table product_sales (product varchar2(10),sale_date date,amount number);

Table created.

SQL> insert into product_sales values ('beer',sysdate-10,10);
SQL> insert into product_sales values ('beer',sysdate-9,2),
SQL> insert into product_sales values ('beer',sysdate-7,100);
SQL> insert into product_sales values ('wine',sysdate-14,10),
SQL> insert into product_sales values ('wine',sysdate-4,1)
SQL> insert into product_sales values ('wine',sysdate-1,11);
```

VALUES constructor… the future

```
SQL> create table product_sales (product varchar2(10),sale_date date,amount number);

Table created.

SQL> insert into product_sales values ('beer',sysdate-10,10),
                                      ('beer',sysdate-9,2),
                                      ('beer',sysdate-7,100);

3 rows created.

SQL> insert into product_sales values ('wine',sysdate-14,10),
                                      ('wine',sysdate-4,1),
                                      ('wine',sysdate-1,11);

3 rows created.
```

VALUES constructor

```
SQL> select *
     from   (values
                (1, 'beer', 'My favourite'),
                (2, 'wine', 'My second best'),
                (3, 'whisky', 'My third choice')
            ) tab (position, name, description);

  POSITION NAME   DESCRIPTION
---------- ------ ---------------
         1 beer   My favourite
         2 wine   My second best
         3 whisky My third choice
```

```
SQL> with ranking(position, name, description) as
        (values
                (1, 'beer', 'My favourite'),
                (2, 'wine', 'My second best'),
                (3, 'whisky', 'My third choice')
        )
     select * from ranking;

  POSITION NAME   DESCRIPTION
---------- ------ ---------------
         1 beer   My favourite
         2 wine   My second best
         3 whisky My third choice
```

GROUP BY improvements

```
SQL> select * from product_sales;

PRODUCT    SALE_DATE      AMOUNT
---------- --------- ----------
beer       13-SEP-24          10
beer       14-SEP-24           2
beer       16-SEP-24         100
wine       09-SEP-24          10
wine       19-SEP-24           1
wine       22-SEP-24          11
```

```
SQL> select product,to_char(sale_date,'DAY') weekday,
            sum(amount) sum
     from product_sales
     group by product, to_char(sale_date,'DAY')

PRODUCT    WEEKDAY            SUM
---------- ---------- ----------
beer       FRIDAY             10
beer       SATURDAY            2
beer       MONDAY            100
wine       MONDAY             10
wine       THURSDAY            1
wine       SUNDAY             11
```

GROUP BY improvements

```
SQL> select product, to_char(sale_date,'DAY') weekday, sum(amount) sum
     from product_sales
     group by product,weekday;
```

```
PRODUCT    WEEKDAY             SUM
---------- ---------- ----------
beer       FRIDAY              10
beer       SATURDAY             2
beer       MONDAY             100
wine       MONDAY              10
wine       THURSDAY             1
wine       SUNDAY              11
```

GROUP BY improvements

```
SQL> alter session set group_by_position_enabled=true;

Session altered.

SQL> select product,to_char(sale_date,'DAY') weekday, sum(amount) sum
     from product_sales
     group by group by 1,2;

PRODUCT    WEEKDAY             SUM
---------- ---------- ----------
beer       FRIDAY              10
beer       SATURDAY             2
beer       MONDAY             100
wine       MONDAY              10
wine       THURSDAY             1
wine       SUNDAY              11
```

ERROR at line 1:
ORA-03162: "PRODUCT": must appear in the GROUP BY clause or be used in an
aggregate function as 'group_by_position_enabled' is FALSE
Help: https://docs.oracle.com/error-help/db/ora-03162/

Let’s clean up our table

```
SQL> drop table if exists product_sales;

Table dropped.

SQL> drop table if exists product_sales;

Table dropped.
```

A bit misleading...

If we want to recreate it…

```
SQL> create table if not exists product_sales
                                    (product varchar2(10),sale_date date,amount number);

Table created.

SQL> create table if not exists product_sales
                                    (product varchar2(10),sale_date date,amount number);

Table created.
```

BOOL

```
SQL> create table employees (id number generated always as identity, name varchar2(20), is_active bool, is_retired boolean);

Table created.

SQL> insert into employees (name,is_active,is_retired) values
                          ('Obama',false,true),('Trump',false,false),('Biden',true,false);
3 rows created.

SQL> select * from employees;

        ID  NAME                 IS_ACTIVE   IS_RETIRED
---------- -------------------- ----------- -----------
         1 Obama                FALSE       TRUE
         2 Trump                FALSE       FALSE
         3 Biden                TRUE        FALSE

SQL> select * from employees where is_active or is_retired;

        ID NAME                 IS_ACTIVE   IS_RETIRED
---------- -------------------- ----------- -----------
         1 Obama                FALSE       TRUE
         3 Biden                TRUE        FALSE
```

BOOL

```
SQL> select text, to_boolean(text) to_bool
     from   (values ('yes'),('on'),('true'),('1'),('0'),('off'),('n')) tab (text);


TEXT TO_BOOL
---- -----------
yes  TRUE
on   TRUE
true TRUE
1    TRUE
0    FALSE
off  FALSE
n    FALSE

SQL> select to_boolean('a');
select to_boolean('a')
       *
ERROR at line 1:
ORA-61800: invalid boolean literal: a
Help: https://docs.oracle.com/error-help/db/ora-61800/
```

SCHEMA-LEVEL PERMISSIONS

```
SQL> grant select any table on schema bob to alice;

Grant succeeded.

SQL> insert into bob.secret_table values (1024);

1 row created.

SQL> commit;

Commit complete.
```

```
SQL> connect alice
Connected.

SQL> select * from bob.secret_table;

         A
----------
      1024
```

SCHEMA-LEVEL PERMISSIONS

```
SQL> select * from DBA_SCHEMA_PRIVS

GRANTEE  PRIVILEGE             SCHEMA   ADM COM INH
-------- --------------------- -------- --- --- ---
ALICE    SELECT ANY TABLE      BOB      NO  NO  NO

SQL> grant all privileges on schema alice to bob;

Grant succeeded.

SQL> revoke all privileges on schema alice from bob;

Revoke succeeded.
```

PROPERTY GRAPHS

```
SQL> create table people (person_id number primary key, name varchar2(20));

SQL> create table connections (connection_id number primary key,
       person_id1 number,
       person_id2 number,
       comments varchar2(20));
```

```
SQL> select * from people;

 PERSON_ID NAME
---------- --------------------
         1 Freddie Mercury
         2 Mary Austin
         3 Brian May
```

```
SQL> select * from connections;

CONNECTION_ID PERSON_ID1 PERSON_ID2 COMMENTS
------------- ---------- ---------- ---------
            1          1          2 Partners
            2          1          3 Friends
```

PROPERTY GRAPHS

```
SQL> create property graph connections_pg
       vertex tables (
         people
           key (person_id)
           label person
           properties all columns
       )
       edge tables (
         connections
           key (connection_id)
           source key (person_id1) references people (person_id)
           destination key (person_id2) references people (person_id)
           label connection
           properties all columns
       );

Property graph created.
```

PROPERTY GRAPHS

```
SQL> select person1, person2, comments
     from   graph_table (connections_pg
                         match
                         (p1 is person) -[c is connection]-> (p2 is person)
                         columns( p1.name as person1,
                                  p2.name as person2,
                                  c.comments as comments)
                    );

PERSON1              PERSON2              COMMENTS
-------------------- -------------------- --------------------
Freddie Mercury      Brian May            Friends
Freddie Mercury      Mary Austin          Partners
```

PROPERTY GRAPHS

```
SQL> select person1, comments1, person2, comments2, person3
     from   graph_table (connections_pg
         match
         (p1 is person where p1.name='Brian May') -[c1 is connection]-
             (p2 is person) -[c2 is connection]-
                    (p3 is person where p3.name='Mary Austin')
         columns (p1.name as person1,
                  p2.name as person2,
                  p3.name as person3,
                  c1.comments as comments1,
                  c2.comments as comments2)
       );

PERSON1      COMMENTS1   PERSON2          COMMENTS2   PERSON3
------------ ----------- ---------------- ----------- -------------
Brian May    Friends     Freddie Mercury  Partners    Mary Austin
```

Staging tables

```
SQL> create table staging_table (
       id number,
       name varchar2(30),
       city varchar2(30),
       country varchar2(30)
     ) for staging;
SQL> insert into staging_table values (1,'John Smith','Zurich','CH'),
       (2,'Jane Doe','Zuerich','Switzerland'), (3,'James Cook','Zurich','Swiss');
SQL> update staging_table set city='Zurich';
SQL> update staging_table set country='Switzerland';
SQL> select * from staging_table

   ID NAME            CITY      COUNTRY
----- --------------- --------- -----------
    1 John Smith      Zurich    Switzerland
    2 Jane Doe        Zurich    Switzerland
    3 James Cook      Zurich    Switzerland
SQL> insert into other_table select * from staging_table;
SQL> truncate table staging_table;
```

No stats!
only dynamic sampling

Direct JOINs for UPDATE/DELETE

```
SQL> update  t1 a
     set     a.xxx = 7
     from    t2 b
     where   a.id = b.id
         and b.id <= 5;

SQL> delete  t1 a
     from    t2 b
     where   a.id = b.id
         and b.id <= 5;
```

https://medium.com/@andrei.manoliu/update-and-delete-statements-f68990cfa997

PL/SQL transpiler

The SQL Transpiler automatically and wherever possible converts (transpiles) PL/SQL functions within SQL into SQL expressions, without user intervention.

The SQL Transpiler is disabled by default.

```
SQL_TRANSPILER = [ON | OFF]
```

Helps avoid PL/SQL vs SQL context switches
a lot of limitations e.g. doesn't work for packages

```
create function get_month_abbreviation ( date_value date ) return varchar2 is
begin
  return to_char ( date_value, 'MON', 'NLS_DATE_LANGUAGE=English' );
end;
/

select employee_id from hr.employees where get_month_abbreviation( hire_date ) = 'MAY';

Predicate Information (identified by operation id):
filter(TO_CHAR(INTERNAL_FUNCTION("HIRE_DATE"),'MON','NLS_DATE_LANGUAGE=English')='MAY')
```

https://docs.oracle.com/en/database/oracle/oracle-database/23/tgsql/introduction-to-sql-tuning.html

JSON-Relational duality

```
SQL> create or replace json relational duality view department_dv as
       select json {'_id' : d.deptno,
                    'departmentName'   : d.dname,
                    'location'         : d.loc,
                    'employees' :
                      [ select json {'employeeNumber' : e.empno,
                                     'employeeName'   : e.ename,
                                     'job'            : e.job,
                                     'salary'         : e.sal}
                        from   emp e with insert update delete
                        where  d.deptno = e.deptno ]}
from dept d with insert update delete;
```

https://oracle-base.com/articles/23/json-relational-duality-views-23

JSON-Relational duality

```
SQL> select json_serialize(d.data pretty)
     from   department_dv d
     where  d.data."_id" = 40;

JSON_SERIALIZE(D.DATAPRETTY)
--------------------------------------------------------------------------------
{
  "_id" : 40,
  "_metadata" :
  {
    "etag" : "6FAB9798FF405D87F0EB44456398A5D5",
    "asof" : "00000000002F2799"
  },
  "departmentName" : "OPERATIONS",
  "location" : "BOSTON",
  "employees" :
  [
  ]
}
```

Optimistic locking based on the etag

https://oracle-base.com/articles/23/json-relational-duality-views-23

JSON-Relational duality

```
SQL> insert into department_dv d (data)
values ('
{
  "_id" : 50,
  "departmentName" : "DBA",
  "location" : "BIRMINGHAM",
  "employees" : [
    {
      "employeeNumber" : 9999,
      "employeeName" : "HALL",
      "job" : "CLERK",
      "salary" : 500
    }
  ]
}');
```

```
SQL> select * from dept where deptno = 50;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        50 DBA            BIRMINGHAM


SQL> select empno, ename, job
     from   emp
     where  deptno = 50;

     EMPNO ENAME      JOB
---------- ---------- ---------
      9999 HALL       CLERK
```

https://oracle-base.com/articles/23/json-relational-duality-views-23

Lots of JSON improvements

JSON_ARRAY accepts subquery as input:

```
json_array(select json_object('employee-number' : e.empno) from emp e)
```

`JSON_BEHAVIOR` **parameter controls behaviour of error handling**
`(ERROR ON ERROR, NULL ON ERROR, FALSE ON ERROR, etc.)`

Many improvements to JSON data type

JSON Schema to check validity and structure of JSON documents – `VALIDATE` **keyword**

`DBMS_JSON.JSON_TYPE_CONVERTIBLE_CHECK` **to help migration CLOB -> JSON**

DEFAULT ON NULL FOR INSERT AND UPDATE

```
SQL> create table t1(
       id number,
       description varchar2(15) default 'banana'
       description1 varchar2(15) default on null 'pear',
       description2 varchar2(15) default on null for insert only 'apple',
       description3 varchar2(15) default on null for insert and update 'kiwi'
);
```

https://oracle-base.com/articles/23/default-on-null-for-insert-and-update-23

Priority of Transactions

```
SQL> ALTER SESSION SET txn_priority = 'HIGH/MEDIUM/LOW';
SQL> ALTER SESSION SET priority_txns_high_wait_target = 30;
SQL> ALTER SESSION SET priority_txns_medium_wait_target = 90;
SQL> ALTER SESSION SET priority_txns_mode = 'TRACK/ROLLBACK';
```

If a HIGH priority transaction is blocked for a row lock, Oracle Database can roll back the transaction that is holding the row lock only if the holder is LOW or MEDIUM priority. Oracle Database never rolls back a HIGH priority transaction.

If a MEDIUM priority transaction is blocked for a row lock, Oracle Database can roll back the transaction that is holding the row lock only if the holder is LOW priority.

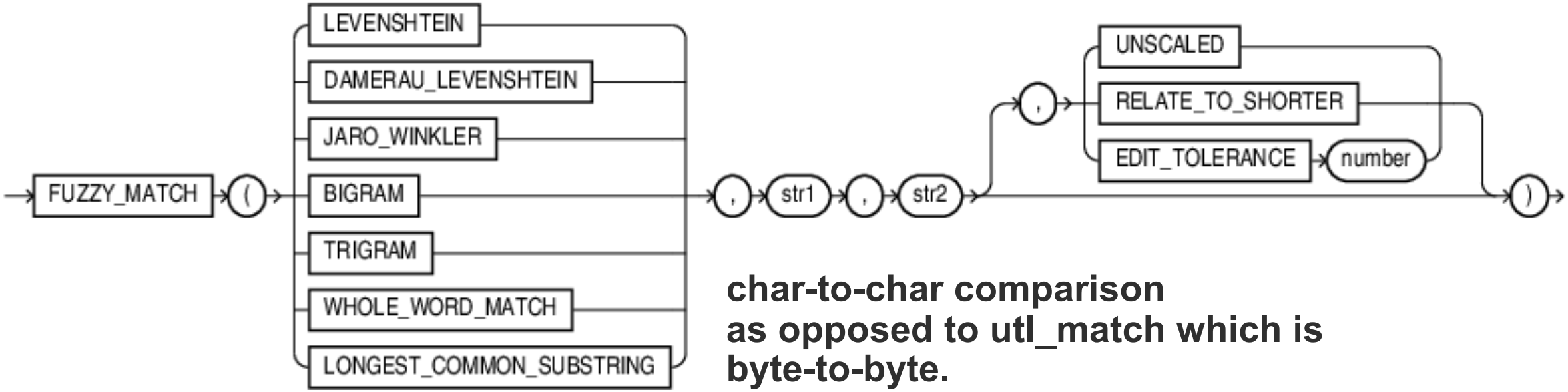
If a LOW priority transaction is blocked for a row lock, Oracle Database will not attempt to roll back the transaction holding the row lock irrespective of its priority.

https://blogs.oracle.com/dbstorage/post/new-priority-transactions-capability-with-oracle-database-23ai

FUZZY_MATCH & PHONIC_ENCODE

```
SQL> select fuzzy_match(levenshtein,'Andrzej Nowicki','Andrei Nowitzki');

FUZZY_MATCH(LEVENSHTEIN,'ANDRZEJNOWICKI','ANDREINOWITZKI')
----------------------------------------------------------
                                                        74
```



char-to-char comparison as opposed to utl_match which is byte-to-byte.

https://docs.oracle.com/en/database/oracle/oracle-database/23/sqlrf/data-quality-operators.html

BIGFILE TABLESPACES

Starting with Oracle Database 23ai, the SYSTEM, SYSAUX, and USER tablespaces are created as bigfile tablespaces by default.
A database upgraded from a previous release retains its tablespace type.

https://docs.oracle.com/en/database/oracle/oracle-database/23/cncpt/logical-storage-structures.html

BIGFILE TABLESPACES – let's check

```
SQL> alter session set
     container=cdb$root;

Session altered.

SQL> select tablespace_name, bigfile
     from dba_tablespaces;

TABLESPACE_NAME                BIG
------------------------------ ---
SYSTEM                         YES
SYSAUX                         YES
UNDOTBS1                       YES
TEMP                           NO
USERS                          YES
```

```
SQL> alter session set
     container=freepdb1;

Session altered.

SQL> select tablespace_name, bigfile
     from dba_tablespaces;

TABLESPACE_NAME                BIG
------------------------------ ---
SYSTEM                         YES
SYSAUX                         YES
UNDOTBS1                       YES
TEMP                           NO
USERS                          NO
```

VECTOR

```
SQL> exec dbms_vector.load_onnx_model(
             directory=>'model_dir',
             file_name => 'all_MiniLM_L12_v2.onnx',
             model_name => 'ALL_MINILM_L12_V2',
             metadata => JSON('{
                                 "function" : "embedding",
                                 "embeddingOutput" : "embedding",
                                 "input": {"input": ["DATA"]}
                               } ')
             );
```

VECTOR

```
SQL> SELECT VECTOR_EMBEDDING(
                                ALL_MINILM_L12_V2
                                USING 'The quick brown fox jumped' as DATA
                              ) AS embedding;

EMBEDDING
--------------------------------------------------------------------------------
[1.655170S1E-002,3.19098569E-002,-1.96293015E-002,-3.56926955E-002,9.21710581E-0
```

VECTOR

```
SQL> desc vector.my_data
 Name                 Null?    Type
 -------------------- -------- ----------------------------
 ID                            NUMBER
 BEER_NAME                     VARCHAR2(128)
 INFO                          VARCHAR2(4000)

SQL> alter table vector.my_data add vector_l12_v2 vector;
SQL> desc vector.my_data

 Name                 Null?    Type
 -------------------- -------- ----------------------------
 ID                            NUMBER
 BEER_NAME                     VARCHAR2(128)
 INFO                          VARCHAR2(4000)
 VECTOR_L12_V2                 CLOB VALUE
```

VECTOR INDEX

```
SQL> create vector index vector.l12_v2a on vector.my_data(vector_l12_v2)
             ORGANIZATION INMEMORY NEIGHBOR GRAPH
             distance cosine
             with target accuracy 95;
```

VECTOR SEARCH

```
SQL>   select beer_name, info
       from vector.my_data
       order by vector_distance(
                    VECTOR_L12_V2,
                    VECTOR_EMBEDDING(ALL_MINILM_L12_V2 USING
                                     '&prompt' as data),
                    cosine)
       fetch approximate first 5 rows only;
```

VECTOR SEARCH

Prompt: 'Fruity with some tangerine aftertaste'

Commodore Perry IPA
A medium-bodied and well hopped India Pale Ale with a dry, fruity aftertaste.

Dreadnaught Imperial IPA
A hop lovera??s dream! Mango and peach aromas with a crisp citrus finish.

Pays du Soleil
Amber ale with Palmetto berries and hibiscus.

Bam Noire
Dark, smooth, delicious. Aromas of worn leather and cool autumn nights. Notes of sweet plum and toasted raisin, hints of coffee and cacao. Lingering tart and refreshing finish. Only available for a few short months. Not to be missed.

Broadside Ale
Rich fruitcake aromas a?? almonds, zest and conserved fruit. A wonderful balance of malt and hop flavours. A beer to savour and rich in flavour.

VECTOR SEARCH – quick demo

```
SQL>   select beer_name, info
       from vector.my_data
       order by vector_distance(
                    VECTOR_L12_V2,
                    VECTOR_EMBEDDING(ALL_MINILM_L12_V2 USING
                                         '&prompt' as data),
                    cosine)
       fetch approximate first 5 rows only;
```

23ai
PL/SQL packages

DBMS_HCHECK

Replacing the hcheck.sql script which checks the internal dictionary structures

It didn’t work for me… package doesn’t exist

But people on blogs managed to run it.

DBMS_SEARCH

Indexing of multiple schema objects in a single index.

```
exec DBMS_SEARCH.CREATE_INDEX('MYINDEX');
exec DBMS_SEARCH.ADD_SOURCE('MYINDEX','PRODUCTS');
exec DBMS_SEARCH.ADD_SOURCE('MYINDEX','CUSTOMERS’);

SELECT METADATA
from MYINDEX
WHERE CONTAINS(data,'shiny or street')>0;

METADATA
--------------------------------------------------------------------------------
{"OWNER":"SCOTT","SOURCE":"PRODUCTS","KEY":{"ID":2}}
{"OWNER":"SCOTT","SOURCE":"CUSTOMERS","KEY":{"ID":1}}
{"OWNER":"SCOTT","SOURCE":"CUSTOMERS","KEY":{"ID":99}}
```

https://docs.oracle.com/en/database/oracle/oracle-database/23/ccapp/performing-ubiquitous-search-dbms_search-apis.html

DBMS_SQL_FIREWALL

SQL Firewall – database firewall features such as allow-listing, deny-listing, and object- and command-based access control inside the Oracle Database kernel.

Should help preventing SQL Injections

https://docs.oracle.com/en/database/oracle/oracle-database/23/arpls/dbms_sql_firewall.html

Honourable mentions

RETURNING clause returns new/old values

Stored procedures in JavaScript

SQL Domains – reuse the same constraints checks across different objects

Object Annotations – comments on steroids

Lock-free reservations – for modification of rows in highly-concurrent fashion

Max columns limit raised to 4096: `compatible = 23.0.0, max_columns=EXTENDED`

New powerful role: `db_developer_role`

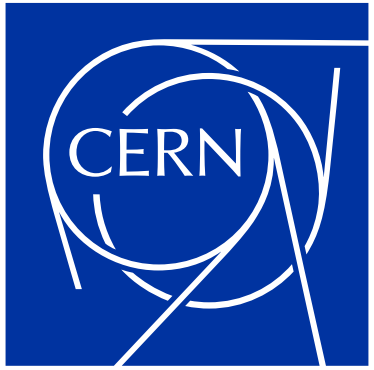
Column level auditing

References

Oracle Docs	https://docs.oracle.com/en/database/oracle/oracle-database/index.html
Tim Hall's Articles	https://oracle-base.com/articles/23/articles-23
Blogs Oracle	https://blogs.oracle.com/database/


Come visit CERN!
https://visit.cern

Thank you !



andrzejnowicki

andrzej.nowicki@cern.ch

www.andrzejnowicki.pl

Slides are available: